UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

(2)

AD-A215 481

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

ELECTE
DEC 07 1989
S B D

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 89-1606
6a. NAME OF PERFORMING ORGANIZATION American Institute of Aeronautics and Astronautics 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research
6c. ADDRESS (City, State, and ZIP Code) 370 L'Enfant Promenade, S. W. Washington, DC 20024-2518	7b. ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR 8b. OFFICE SYMBOL (If applicable) NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-87-0159
8c. ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61102F PROJECT NO. 2304 TASK NO. A7 WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification) (AIAA-87-1655)
SECOND AIAA/NASA/USAF SYMPOSIUM ON AUTOMATION, ROBOTICS AND ADVANCED COMPUTING FOR THE NATIONAL SPACE PROGRAM (Sub-titles continued on reverse of this form.)

12. PERSONAL AUTHOR(S)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
FINAL	FROM 1 Mar 87 TO 28 Feb 88		

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	[illegible]

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The U.S. Navy has a wide spectrum of applications to which AI can be applied, including manufacturing and logistics, and operational applications in surface ships, submarines, aircraft and space applications. The Navy, through the Office of Naval Research (ONR), has supported artificial intelligence research from its earliest emergence as a discipline, and continues to be an important contributor to university-based artificial intelligence research. Increasingly the Navy is conducting applied research programs within its own laboratories with the goal of transitioning the technology into service. These programs cover a range of application areas, many of which are closely related to civilian problems, but often they exhibit a unique military flavor. Applications such as fault diagnosis, inspection, planning aids and image analysis clearly have close counterparts in the commercial world. Target classification, battle management, naval message processing and enemy plan recognition are unlike their closest commercial counterparts in a number of respects, particularly when operational impact, real-time operation and computing power limitations are considered. Navy projects in several of these areas have matured to the degree that full scale utilization of the technology could begin within three to five years.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr Abraham Waksman	22b. TELEPHONE (Include Area Code) (202) 767-5027 22c. OFFICE SYMBOL NM

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

89 12 05 069

II. (Continued) Sub-titles:

AIAA-87-1660	Applied Artifical Intelligence in Navy
AIAA-87-1661	Issues and Themes in Information Science and Technology
AIAA-87-1665	Vision and Navigation for the Carnegie Mellon Navlab
AIAA-87-1667	The Connection Machine: A Fine Grained Multi-Processor
AIAA-87-1674	Computational Themes in Applications of Visual Perception
AIAA-87-1676	NASA Systems Autonomy Demonstration Project: Development of Space Station Automation Technology
AIAA-87-1678	Hierarchical Classification: Its Usefulness for Diagnosis and Sensor Validation
AIAA-87-1682	AI Applications for Space Support and Satellite Autonomy
AIAA-87-1685	Validation of Knowledge-Based Systems
AIAA-87-1686	New Concepts in Tele-Autonomous Systems
AIAA-87-1687	Software Architecture For Manufacturing and Space Robotics
AIAA-87-1688	Integrated Army Robotics Thrust
AIAA-87-1694	Design and Control of Modular Kinematically-Redundant Manipulators
AIAA-87-1695	NASA's Technology Plans - Will Technology Be Ready When We Are
AIAA-87-1697	Computer Architecture for Future Spacecraft

AFOSR-TR- 89-1606

AIAA-87-1655
Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

D. Myers, NASA Headquarters,
Washington, DC

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program
March 9-11, 1987/Arlington, VA

SECOND AIAA/NASA USAF SYMPOSIUM ON AUTOMATION, ROBOTICS AND ADVANCED COMPUTING FOR THE NATIONAL SPACE PROGRAM

Dale Myers
Deputy Administrator
NASA Headquarters
Washington, DC

Thank you, Noel (Hinners), and good morning to you all. On behalf of all of us at NASA, I'm pleased to welcome you to this most important symposium.

Nearly 40 years ago, one of the fathers of cybernetics, Norbert Weiner, described automation as a "new development which has unbounded possibilities for good and for evil."

Mr. Weiner was right, of course. New technologies are, in themselves, neutral. It is only in our capabilities and intentions to apply them that they take on moral significance. The real importance of advanced technology lies in what we do with it. I am pleased that this Symposium will help us to focus on applying some of mankind's most advanced technologies to create unbounded possibilities for good.

As we prepare to enter the Space Station era, it is clearer than ever that the Station's development will also drive the development of automation, robotics and advanced computing, both in space and on Earth.

Advances in these technologies, to be stimulated by the Space Station Program, will benefit the United States in many ways.

Such advances will increase productivity in space for commerce and science. They will also increase productivity throughout the United States economy, as the new technologies are transferred back to industries on Earth. And they will help to preserve United States' leadership both in space and at the cutting edge of technology on Earth.

NASA welcomes the challenge of developing a highly automated and autonomous Space Station. Indeed, such a facility will be essential, not only to the station's development, but to its evolution and to future space endeavors through the end of the century and beyond.

The successful use of such technologies will ensure that future space missions, be they manned or unmanned, will be beneficial, productive and cost-effective. NASA's goal in developing these technologies is to ensure that we achieve the best mix of people and machines to do our work effectively and efficiently.

If we can meet that goal, we will enhance both the productivity and the safety of the pioneers who will be living, working and learning in space permanently in just a few short years. Indeed, the safety of our people is NASA's highest priority, because people are our most precious resource.

What is NASA doing to achieve that balanced mix of people and machines so essential to future progress?

We have developed a research program designed to exploit automation and robotics technology to the highest degree possible consistent with our resources. The research is carried out by our Office of Aeronautics and Space Technology, and its fruits are transferred to three other NASA program offices; that of Space Flight, of the Space Station and of Space Science and Applications.

Our research program has three objectives. First, we aim to decrease the cost of ground control, and ground processing and checkout; second, we want to increase the capability and flexibility of space operations. Finally, we want to increase the probability of mission success.

Artificial Intelligence Technology will be used to reduce the size of the ground control contingent. Telerobotics will be used to enable increased space assembly, servicing and repair.

In the long run, we expect the program will have extremely beneficial results.

Its goals are to decrease mission operations manpower by 75 per cent; to replace half [illegible] our extra-vehicular activities with t[illegible]botics; and to enable remote asse[illegible], servicing and repair through telerobotics in both geosynchronous Earth orbit and in polar orbit for the Space Shuttle, the Space Station's Orbiting Maneuvering Vehicle and Flight Telerobotics Servicer and other orbiting facilities.

NASA's work is designed to complement the extensive efforts of industry and the military in special areas that several study groups have identified as requiring NASA leadership.

For example, one area in which NASA clearly has taken a leadership role is that of traded control. As you know, this is the ability to switch from human to autonomous control of a machine gracefully and smoothly - something like using cruise control in your car. Another area is that of operator-machine interfaces.

NASA gained this expertise because it has employed automation and robotics in both manned and unmanned missions since the beginning of the space program. Our unmanned planetary explorers and other scientific spacecraft all have had and continue to have highly automated systems. And, as we all know, the planned Mercury flights were so automated that the astronauts felt they would be little more than passengers in their capsules.

Our heritage in robotics was highly visible in the Viking Mars Lander, and remains so in the Shuttle's Remote Manipulator System. Clearly, automation and robotics are not new to NASA.

What is new is our increased focus on advanced automation and robotics technology in recent years. This new surge of interest and activity has come about because of two related government initiatives.

The first is the Space Station program, initiated by President Reagan in his 1984 State of the Union Address. The second is high Congressional interest. This was manifested in July 1984 when Congress passed legislation, now known as Public Law 98-371, which directed NASA to put the best brains in the country to work to identify and develop space station systems which advance automation and robotics technologies, not only on the initial Space Station, but throughout its evolution.

That same public law spawned the formation of two groups. The first is the NASA's Advanced Technology Advisory Committee, a group of NASA experts formed to make the reports the law made mandatory. The second is the Automation and Robotics Panel, composed of non-NASA experts from industry government and academia. Both of these groups have made and continue to make valuable contributions to the NASA program.

Their advice and assistance has been invaluable in developing our agency-wide focused and augmented automation and robotics program.

We are developing the program, as we develop any major program, based on strict assignments of roles and responsibilities and careful planning assumptions. Indeed, we had ten planning assumptions, and they have served as our Ten Commandments as the program takes shape.

Because they are so important, I would like to summarize those planning assumptions now.

The first is that there will be two foci: telerobotics and system autonomy, or expert systems.

Second, each focus will have a series of ground demonstrations of an evolutionary testbed to show increasing capability of integrated technologies.

Third, there will be a core technology program to develop the capabilities needed to enable the demonstration sequences.

Fourth, the resources balance will be 2/3 for core technology and 1/3 for the demonstration sequences.

Fifth, any flight demonstrations will be funded in conjunction with the NASA User Offices, that is, the Office of the Space Station and the Office of Space Flight.

Sixth, each of the two ground demonstration sequences will take place at a NASA Center, but not necessarily the same center.

Seventh, the core technologies will be developed at various sites.

Eighth, government, industry and university research will be leveraged by using 40 percent of the program funding to sponsor research outside of NASA.

Ninth, for each focus, the technology development will begin with an early demonstration of current capabilities and move forward in aggressive, but reasonable increments, leading to the development of a revolutionary technology for both automation and robotics.

Finally, arrangements will be made to link research centers, user centers, NASA program offices and appropriate universities and industries into working teams.

Using the approach I have just outlined, NASA is moving vigorously to integrate application of advanced automation and robotics technology into the development of the Space Station and into the Space Transportation System as a whole. Our most critical needs for those technologies for the Station will be in the initial assembly and building phase; for autonomous operations, once it is assembled and built; and in the efficient servicing of the Station, payloads and satellites.

It is obvious that our contractor's cooperation is vital in this program. That is why we have made it clear to them, both in the Phase B definition and preliminary design phase of the program, and as we approach the developmental phase, that automation and robotics technology is required for the Space Station. Indeed, we will evaluate contractor proposals for the final design and fabrication phase of the program for their responsiveness to automation and robotics requirements.

I am pleased that our contractors are coming through very well. They identified and recommended automation and robotics applications during Phase B, and have amplified those plans in innovative and skillful ways as we prepare for development in Phases C and D.

Funding for NASA's automation and robotics research and technology program is growing incrementally every year. It has gone from $10.2 million at the program's inception in FY 1986 to a proposed $25.7 million for Fiscal 1988.

I am pleased to report that funds have been committed and significant progress has been made in developing a flight telerobot servicer. This machine will be available as the first element launch of the Space Station and will be used to support assembly, maintenance, and servicing of the station.

Research and development also is being conducted throughout NASA and by its contractors on expert systems to support Space Station operations and scientific research and manufacturing in space.

The university community is on board, as well. NASA has selected the University of Wisconsin to establish a Center for the Commercial Development of Space, specializing in space automation and robotics.

As we move forward with these activities it is clear that potential applications of this work will be important not only for the Space Station program, but for the Space Transportation System as a whole and for industries on Earth.

Indeed, given the long-term benefits in greater Space Station productivity and in major productivity improvements that would result from transfer of expert systems to industries on Earth, many believe that NASA should make a greater investment in automation and robotics technology.

For example, the recent report of the Automation and Robotics Task Force commissioned by NASA's Space Systems and Technology Advisory Committee estimates that the program is underfunded by a factor of 2 to 4. The Task Force asserted that to be successful, and to do all that NASA should be doing in this promising area, funding should be increased substantially.

NASA agrees with that assessment. And I can only say that if the Congress will give us the resources, we will do the best job that can be done.

Our great adventure in space has hardly begun. The Space Station and the missions to follow will challenge our ingenuity and skills to the fullest. But from this effort will come new knowledge and new technological breakthroughs we can hardly imagine today.

Our challenge is to make the most of our opportunities to benefit ourselves and future generations.

Last month, President Reagan spoke to the NASA employees on the first anniversary of the Challenger accident. He said: "Whether it is the exploration of space or the applications of space research here at home, the future to which you are leading us is bright, the challenge which you are shouldering for all mankind is one that we cannot turn away from."

NASA welcomes that challenge. We will never turn away from it. We will continue to push ahead, with dedication and resolve, toward a brighter future for all mankind.

Thank you very much.

AIAA-87-1660
Applied Artificial Intelligence in Navy

R. P. Shumaker, Naval
Research Laboratory,
Washington, DC

**Second AIAA/NASA/USAF Symposium on
Automation, Robotics and Advanced Computing
for the National Space Program**

March 9-11, 1987/Arlington, VA

APPLIED ARTIFICIAL INTELLIGENCE IN NAVY

Randall P. Shumaker
Naval Research Laboratory
Washington, DC

Abstract

The U.S. Navy has a wide spectrum of applications to which AI can be applied, including manufacturing and logistics, and operational applications in surface ships, submarines, aircraft and space applications. The Navy, through the Office of Naval Research (ONR), has supported artificial intelligence research from its earliest emergence as a discipline, and continues to be an important contributor to university-based artificial intelligence research. Increasingly the Navy is conducting applied research programs within its own laboratories with the goal of transitioning the technology into service. These programs cover a range of application areas, many of which are closely related to civilian problems, but often they exhibit a unique military flavor. Applications such as fault diagnosis, inspection, planning aids and image analysis clearly have close counterparts in the commercial world. Target classification, battle management, naval message processing and enemy plan recognition are unlike their closest commercial counterparts in a number of respects, particularly when operational impact, real-time operation and computing power limitations are considered. Navy projects in several of these areas have matured to the degree that full scale utilization of the technology could begin within three to five years.

AIAA-87-1661
Issues and Themes in Information Science and Technology

S. Amarel, Defense Advanced Research Projects Agency, Arlington, VA

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

March 9-11, 1987/Arlington, VA

TS AND THEMES IN INFORMATION SCIENCE AND TECHNOLOGY

Saul Amarel
Defense Advanced Research Projects Agency
Information Science and Technology Office
Arlington, Virginia 22209-2308

Computing is an extremely dynamic field, and it will continue to have major impact on our society. There are several exciting scientific/technological developments in the field today that promise to accelerate further the impact of computing on defense and on our national economy.

The technology of Very Large Scale Integration (VLSI) and networking has spawned several novel multiprocessor architectures. These architectures promise to provide the very high performance capabilities that are needed in applications such as Vision, Speech, Complex Symbolic Processing and Large Scientific problems. DARPA has a major effort in this area as part of our Strategic Computing Program. Results so far are very encouraging. Much remians to be done, especially in attaining a better understanding of the capabilities of new architectures and in developing effective ways of applying them to the solution of various types of computationally-intensive problems. Basically, we face the challenge of developing appropriate computational paradigms for parallel computing. Another important task is to establish better links between developments in conventional (primarily numerically-oriented) supercomputers and the new explorations in highly parallel architectures.

We are now at a point where Artificial Intelligence (AI) is starting to have noticable impact on a number of industrial and defense problems. DARPA has been the primary source of support for AI research over the last twenty years. Scientific developments in AI promise to elucidate the nature of intelligent action in man and machine; in addition, they promise to bring about a new generation of machine intelligence technology that will extend considerably the scope and power of computing that will be available to various classes of users. The major thrust of the DARPA Strategic Computing Program is to accelerate in a substantial way the development of a machine intelligence technology that can have a significant impact on defense systems and that can strengthen our industrial capabilities. Much work in AI is now directed to the exploration of methods and systems for solving very complex "real life" problems. In parallel with these efforts, we need to pursue basic research on problems of representation, reasoning, learning and discovery, and on frameworks for designing and implementing AI systems.

This country, and the DoD in particular, has an enormous investment in software. This is an area where better methodologies, tools and theories are sorely needed. We must develop much more effective ways of designing, building and maintaining software. AI methods and cost-effective, massive, computer power will certainly help us in this enterprise. In addition, we need to develop better theoretical underpinnings and more powerful automated processes to support the engineering of complex, testable and reliable software.

We have made enormous progress in the area of computer networking and distributed computing. We must build on top of our experience with ARPANET and internetting to provide a more convenient network access to the national (and international) scientific and engineering communities. We can expect completely new patterns of collaborative research and engineering to develop because of the growth and maturation of computer networking technology. Experience to date suggests that these developments will have significant impact on the productivity of professionals. More research is needed on wide-band networks and on reconfigurable networks. Also, we are still facing difficult problems in the design and use of distributed computing systems.

In recent years there has been substantial progress in the application of computers to design and manufacturing. This is an enormously promising area, and it has great significance for the national economy. Most of the work so far has concentrated on the use of computers to design computers and their parts (e.g., VLSI design, digital circuit design, system configuration). It is becoming apparent that many of the current techniques can be extended to other types of design, including the design of mechanical parts and processes. Computer science and technology is now at a point where it can provide the intellectual basis and the tools for the development of a science of design. I believe that the entire area of exploring computing as a basis for assisting and amplifying industrial prodctivity is an excellent candidate for a major national initiative.

Even though the U.S. pioneered many of the early developments in robotics, it is now lagging (relative to Japan) in the production and use of robotics. We now have many of the elements that are needed for the construction of advanced robots, i.e., robots that can effectively integrate perception, planning and action in the pursuit of given goals. At DARPA we have a growing program in this area, including the Autonomous Land Vehicle project. This is another area that may be ready for a major national thrust.

There has been progress in computer-aided instruction and training in the last decade but much more needs to be done in this area. By increasing the effectiveness of computer systems as adjuncts to instruction we can improve the human potential of the nation. This is an extremely important objective. We need additional research in this area, in particular work on effective and adaptive man-machine interfaces.

We are moving to a point where the presence of computers is influencing in a substantial way the approach to many scientific and engineering problems. Basic paradigms for work in physics, in mathematics and in other areas are being reshaped because of the availability of vastly increased amounts of computer power as well as of improved methodologies for harnessing raw computer power to the needs of users. It is important for computer science and technology to be aware of these developments and to try to understand them. This has implications on education in computer science and on the need for cross-disciplinary research.

Computer Science is continuing to suffer from a manpower shortage. It is essential to build university environments that encourage an increased production of graduate students in the discipline. This has implications on the continuity of research support and the availability of up-to-date computational facilities in academia.

We need to maintain an adequate growth rate of computer science research funding in the nation. At DARPA, computing is an area of high priority, and it is being funded at a level which is roughly one quarter of the agency's total budget.

It would be desirable to develop a national policy for computer science research. In order to move towards such a policy, it is important to increase the coordination of planning in computer science among government agencies. The FCCSET committee activities are a move in the right direction. Also, it would be useful to establish a forum for the interchange of ideas on long-range planning with participation from government, industry and academia. The creation of the Computer Science and Technology Board by the National Research Council is an important step in that direction.

Computer science and technology are in a healthy state of development today. There are many research and engineering challenges in the field; there are also many opportunities. Viewed in a broader context, the computer field is in a pivotal position of influence vis-a-vis all other areas of science and advanced technology in the U.S. It is essential that we maintain our momentum in advanced computing in order to maintain/increase the chances of national leadership in other key areas of science and technology, such as Space.

AIAA-87-1665

Vision and Navigation for the Carnegie Mellon Navlab

C. Thorpe, S. Shafer and T. Kanade,
Carnegie Mellon Univ.,
Pittsburgh, PA

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

March 9-11, 1987/Arlington, VA

Vision and Navigation for the Carnegie Mellon Navlab

Charles Thorpe
Steven Shafer
Takeo Kanade
and the members of
the Strategic Computing Vision Lab

1. Introduction

Robotics is where Artificial Intelligence meets the real world. AI deals with symbols, rules, and abstractions, reasoning about concepts and relationships. The real world, in contrast, is tangible, full of exceptions to the rules, and often stubbornly difficult to reduce to logical expressions. Robots must span that gap. They live in the real world, and must sense, move, and manipulate real objects. Yet to be intelligent, they must also reason symbolically. The gap is especially pronounced in the case of outdoor mobile robots. The outdoors is constantly changing, due to wind in trees, changing sun positions, even due to a robot's own tracks from previous runs. And mobility means that a robot is always encountering new and unexpected events. So static models or preloaded maps are inadequate to represent the robot's world.

The tools a robot uses to bridge the chasm between the external world and its internal representation include sensors, image understanding to interpret sensed data, geometrical reasoning, and a concept of time and of the vehicle's motion over time. We are studying those issues by building a mobile robot, the CMU Navlab, and giving it methods of understanding the world. The Navlab has perception routines for understanding color video images and for interpreting range data. CODGER, our "whiteboard", which was developed for the Navlab and its smaller cousin the Terregator, handles much of the modeling of time and geometry. Our architecture coordinates control and information flow between the high-level symbolic processes running on general purpose computers, and the lower-level control running on dedicated real-time hardware. The system built from these tools is now capable of driving the Navlab along narrow asphalt paths near campus while avoiding trees and pausing for joggers that get in its way.

This report describes the Navlab [11] and the software we have built over the past year: color vision, for finding and following roads [12]; 3D perception, for obstacle avoidance [4]; and the CODGER whiteboard [10].

2. The Navlab

The Navlab (short for Navigation Laboratory) is a van converted into a robot, capable of being driven conventionally or under computer control (figure 1). It is self contained, carrying its own sensors, processors, power, and even researchers.

Our previous mobile robots, such as the Terregator, have been reliable workhorses for small-scale experiments. However, we have started to outgrow their capabilities. As we began to experiment with sensor fusion, smaller robots ran out of space and power for multiple sensors. When we wanted to expand our test areas, communications to a remote computer in the lab became more difficult. And as the experiments became more sophisticated, we found it more productive for the experimenters to test or debug new programs near or in the vehicle, instead in a remotely located laboratory.

Figure 1: The Navlab

Navlab is based on a commercial van chassis, with hydraulic drive and electric steering. Computers can steer and drive the van by electric and hydraulic servos, or a human driver can take over control if necessary. It is even licensed in Pennsylvania, so we can drive it by hand to our remote test sites. Once there, the driver pulls a switch and an onboard microprocessor assumes control of the steering and drive. The Navlab has room for researchers and computers on board, and has enough power and space for all our existing and planned sensors. This gets the researchers close to the experiments, and eliminates the need for video and digital communications with remote computers.

Features of the Navlab include:

- **Onboard computers.** We have five computer racks, one for low-level controllers and power smoothing, one for video distribution, VCRs, communications and miscellaneous equipment, two racks for general-purpose processors (currently Sun workstations), and one for a Warp processor.
- **Onboard researchers.** There is always a safety driver in the driver's seat. There is room for four researchers in the back, with a terminal or workstation for each. An overhead shelf holds video monitors and additional terminals. The researchers can monitor both their programs and the vehicle's motion.
- **Onboard power.** The Navlab carries two 5500 Watt generators, plus power conditioning and battery backup for critical components.
- **Onboard sensors.** Above the cab is a pan mount carrying our laser scanner and a mounting rail for a color TV camera. There will eventually be a separate pan/tilt mount for stereo cameras.
- **Evolving controller.** The first computer controller for the Navlab is adequate for our current needs. It will evolve to do smoother motion control, and to interface to an inertial guidance system and possibly to GPS satellite navigation. The controller will also watch vital signs such as computer temperature and vehicle hydraulic pressure.

3. Color Vision

The Navlab uses color vision, specifically multi-class adaptive color classification, to find and follow roads. Image points are classified into "road" or "non-road" principally on the basis of their color. Since the road is not a uniform color, color classification has to have more than one road model, or class, and more than one non-road class. And since conditions change from time to time and from place to place over the test course, the colors have to adapt. Once the image is classified, the road is found with an area-based voting technique that finds the most likely location for the road in the image.

3.1 Vision Principles for the Real World

We based the development of our vision on the following principles:

Assume variation and change. On sunny days, there are shadowed areas, sunlit areas, and patches with dappled sunlight. On rainy days, there are dry patches and wet patches. Some days, there are wet and dry and sunny and shadowed all in the same image. The road has clean spots and other places covered with leaves or with drips of our own hydraulic fluid. And as the sun goes behind a cloud or as the vehicle turns, lighting conditions change. This means that we need to have more than one road color class and more than one non-road class, that those classes need to adapt to changing conditions, and that we need to process images frequently so the change from one image to the next will be moderate.

Use few geometric parameters. A complete description of the road's shape in an image can be quite complex. The road can bend gently or turn abruptly, can vary in width, and can go up or down hill. However, the more parameters there are, the greater the chance of error in finding those parameters. Small misclassifications in an image could give rise to fairly large errors in perceived road geometry. Furthermore, if all the road parameters can vary, there are ambiguous interpretations: does the road actually rise, or does it instead get wider as it goes? We chose to describe the road with only two free parameters: its orientation and its distance from the vehicle. Road width is fixed, we assume a flat world, and we decree that the road is straight. While none of those assumptions are true over a long stretch of the road, they are nearly true within any one image. And the errors in road position that come from those oversimplifications are balanced by less chance of bad interpretations. If there are a few pixels incorrectly classified as road, the worst our method will do is to find a slightly incorrect road. A method that tries to fit more parameters, on the other hand, may come up with the perfect interpretation of part of the road, but could find an abrupt turn or sudden slope near the bad pixels.

Work in the image. The road can either be found by projecting the road shape into the image and searching in image coordinates, or by back projecting the image onto the ground and searching in world coordinates. The problem with the latter approach comes in projecting the image onto an evenly spaced grid in the world. The points on the world grid close to the vehicle correspond to a big area in the lower part of the image; points farther away may correspond to one or a few pixels near the top. So either there has to be a complex weighting scheme, or some image pixels (those at the top, that project to distant world points) will have more weight than other (lower) points. A few noisy pixels can have a big or a small effect, depending on where in the image they lie. On the other hand, working directly in the image makes it much easier to weight all pixels evenly. We can directly search for the road shape that has the greatest number of road pixels and the least non-road pixels. Moreover, projecting a road shape is much more efficient than back projecting all the image pixels.

Calibrate directly. A complete description of a camera has to include its position and orientation in space; its focal length and aspect ratio; lens effects such as fish-eye distortion; and non-linearities in the optics or sensor. The general calibration problem of trying to measure each of these variables is very difficult. It is much easier, and more accurate, to calibrate the whole system rather than trying to tease apart the individual parameters. The easiest method is to take a picture of a known object and build a lookup table that relates each world point to an image pixel and vice versa. Projecting and back projecting are simply done by table lookup, or table lookup for close by values with simple interpolations. Such a table is straightforward to build and provides good accuracy, and there are no instabilities in the calculations.

Use outside constraints. Even without a map of our test course or an expensive inertial navigation system, we know approximately where the road should be based on the previous image and on vehicle motion. Our "whiteboard" can predict where the road should appear if the road were straight and the vehicle navigation were perfect. Adding a suitable margin for curved roads and sloppy navigation still gives useful limits on where in the image to look for the road.

Test. We tried to run our VCR every day we took the vehicle out, to collect images under as many conditions as possible. We had sunny days, cloudy days, rainy days, leaves on trees, leaves turning color, leaves falling, early morning, noon, after dusk, even a partial solar eclipse. Strategies that worked well on one set of images did not always work on the others. Our philosophy was to collect the toughest images from those sets. We ran our best algorithms and printed the classification results, changed parameters or algorithms, reran the data set, and compared results. This gave us the best chance of being methodical and of not introducing new bugs as we went. When the image processing worked to our satisfaction, we ran simulations in the lab that included the whiteboard, range processing, path planning, and a vehicle simulator, with vision processing stored images and interacting with the rest of the system. When the simulations worked in the lab, we moved them to the vehicle. Only after the simulations worked on the vehicle's computers, and we were sure that all necessary software was on the van, did we go into the field for real tests. Even then, everything didn't work, but there were many fewer bugs than there would have been without the simulations and tests.

3.2 Road Following Algorithm

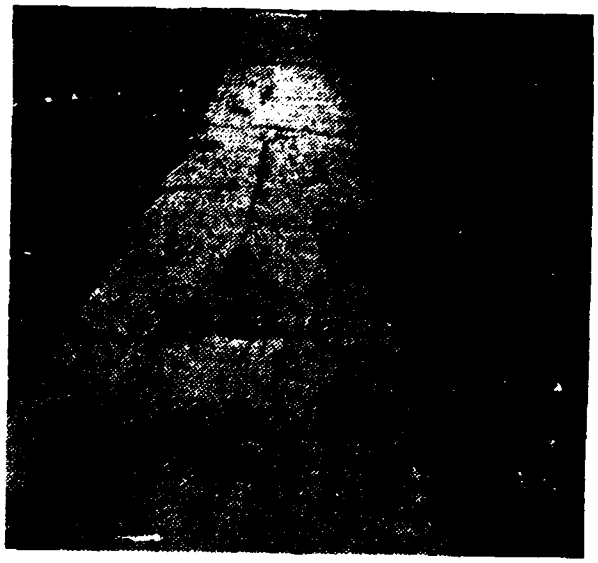


Figure 2: Original Image

We followed those principles in building and tuning adaptive color classification for following roads. Our algorithm involves three stages.

1. Classify each pixel.
2. Use the results of classification to vote for the best-fit road position.
3. Collect new color statistics based on the detected road and nonroad regions.

Pixel classification is done by standard pattern classification. Each class is represented by the means, variances, and covariances of red, green, and blue values, and by its a priori likelihood based on expected fraction of pixels in that class. For each pixel, calculating the class to which it most likely belongs involves finding how far the pixel's values lie from the mean of each class, where distance is measured in standard deviations of that class. Figures 3 and 4 show how each pixel is classified and how well it matches.

Once each point has been classified, we have to find the most likely location of the road. We assume the road is locally straight, and can be described by two parameters (figure 5):

1. The image column of the road's "vanishing point", where the road intercepts the horizon. This gives the road's direction relative to the vehicle.
2. The orientation of the road in the image, which gives how far the vehicle is to the right or left of the centerline.

We set up a two dimensional parameter space, with intercept as one dimension and orientation as the other. Each point classified as road votes for all road intercept / orientation combinations to which it could belong, as shown in figure 6. The orientation / intercept pair that receives the most votes is the one that contains the most road points, and is reported as the road (figures 7 and 8).

Once the road has been found in an image, the color statistics are recalculated for each class (figure 9). The updated color statistics will gradually change as the vehicle moves into a different color road, or as lighting conditions change, or as the color of the surrounding grass, dirt, and trees vary. As long as the processing time per image is low enough that there is a large overlap between images, the statistics adapt as the vehicle moves. The road is picked out by hand in the first image. Thereafter, the process is automatic, using the segmentation from each image to calculate color statistics for the next.

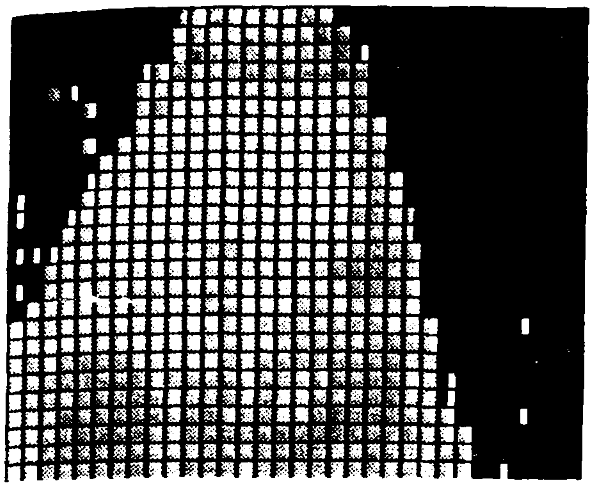


Figure 3: Segmented image. Color and texture cues are used to label points below the horizon into two road and two offroad classes

Figure 4: Road probability image. The pixels that best match typical road colors are displayed brightest.

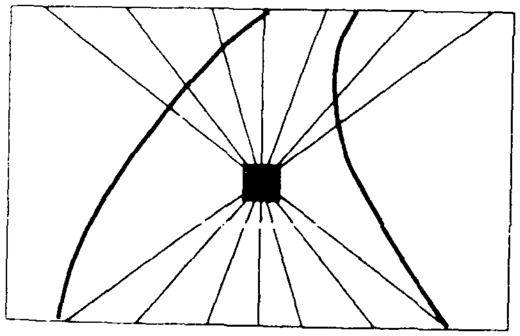


Figure 5: A point classified as road could be a part of roads with different orientations and vanishing points.

There are several variations on this basic theme. One variation is to smooth the images first. This throws out outliers, and tightens the road and nonroad clusters. Another is to have more than one class for road and for non-road, for instance one for wet road and one for dry, or one for shadows and one for sun. If we knew where the wet road and dry road patches were, we could collect separate color statistics for those two classes. Since we don't know where those areas are, we collect new statistics by starting with one class for the top part of the road image and one for the bottom. Using color statistics from those regions, we reclassify the road pixels in the same

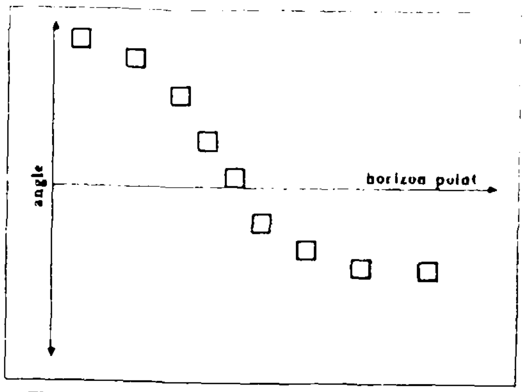


Figure 6: The point from figure 5 would vote for these orientation / intercept values in the parameter space.

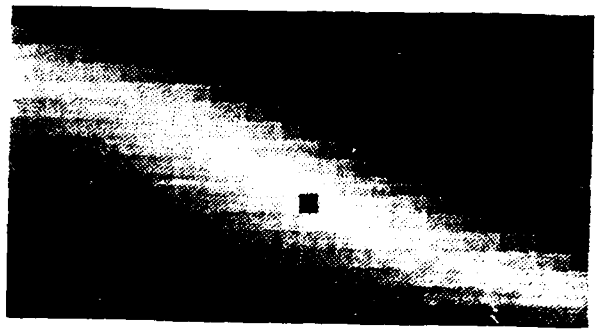


Figure 7: Votes for best road orientation and intercept, and point with most votes.

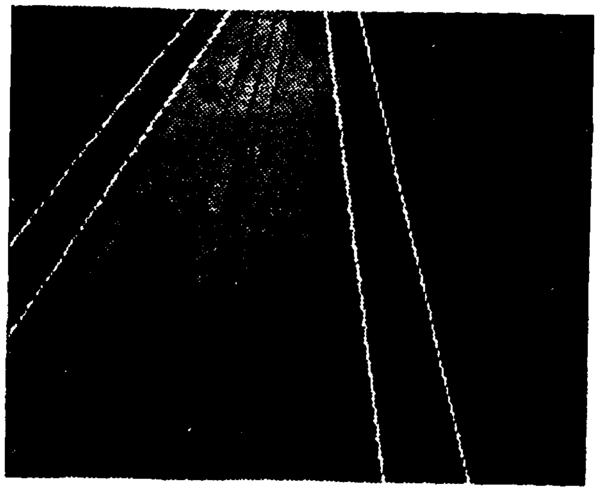


Figure 8: Detected road.

image, and recalculate statistics based on this reclassification. We loop through the classify - update cycle several times for each image. As long as the first two point sets have different distributions, the resulting classes will move towards separate, tighter clusters.

Other variations change the voting for best road. Besides adding votes for road pixels, we subtract votes for non-road points. Votes are weighted according to how well each point matches road or nonroad classes.

There are other clues in an image besides color, for instance visual texture. Roads tend to be smooth, with less high-frequency variation than grass or leaves, as shown in figure 10. Our first texture algorithm ran a Robert's gradient operator, thresholded the result, and counted the number of pixels above threshold in each 16 by 16 block. This method fails to deal adequately with shadow edges, to which Robert's operator responds strongly, or with the interiors of shadows, in which the absolute values of gradients are small because the pixel values are small. The shadow edge problem can be handled by dividing the high frequency gradient by a lower frequency gradient obtained by using Roberts operator on a reduced resolution image. Grass and leaf texture will show up only at high resolution, while shadow edges show up at both. Shadow interiors can be normalized by dividing by the average pixel intensity. We calculate a normalized gradient by

$$\text{normalized gradient} = \frac{\text{high freq gradient}}{\text{low freq gradient} + 3 \cdot \text{mean pixel value}}$$

We threshold the normalized gradient and count the number of pixels above threshold in each block (figure 11).

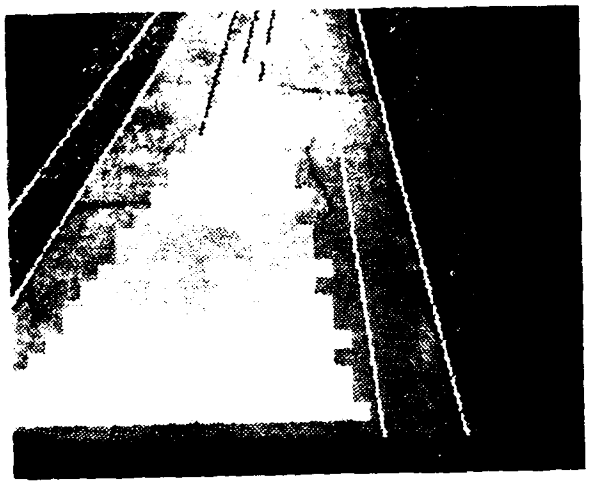


Figure 9: Updating road and nonroad model colors, leaving a safety zone around the detected road region.

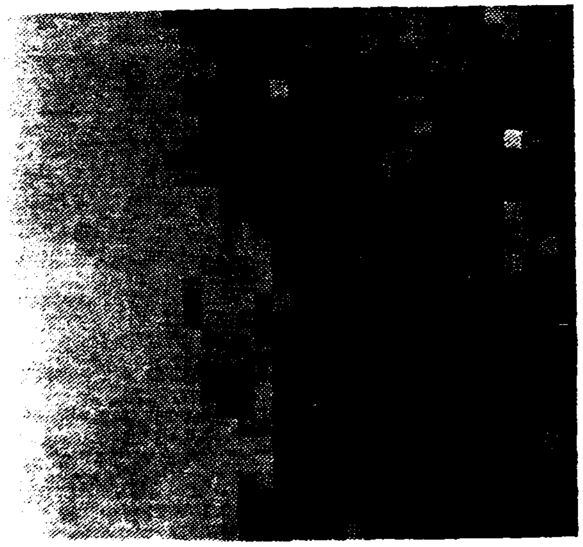


Figure 10: *Zoomed picture of road-nonroad boundary. The road (at left) is much less textured than the grass (at right).*

3.3 Implementation and Results

The best combination we have found is to have two road classes and two non-road classes, and to smooth the image with a 16 by 16 averaging filter. The two road classes handle sunny and shaded road on bright days, or wet and dry patches under overcast conditions. The two non road classes often converge on trees and grass, or grass and dry leaves. Filtering the images by averaging reduces the effect of cracks in the sidewalk and other scene anomalies, and also smooths out color aberrations from misaligned camera guns and from noise in the digitizers and color splitter.

The texture information could be used as a fourth element in the classification along with red, green, and blue. We found texture used this way to be less reliable than color. Instead, we treat texture as a separate feature with fixed statistics (mean and variance for road and nonroad) used in a post processing stage. We calculate road and nonroad probabilities based on color and separately based on texture. We then combine the two with color weighted much more heavily, but texture discriminating in borderline cases.

This algorithm correctly classifies over 95% of our test images in the lab. When we run on the Navlab, with predictions of where the road should appear, our failure rate is very close to 0. The occasional remaining problems come from having too many leaves on the road, so the color statistics of one road class and of nonroad are very close. Not every image is classified perfectly, but almost all are good enough for navigation. We leave a 50 pixel "safety zone" (about two feet on the ground near the vehicle) along the road border that is not used in updating color statistics, to keep from being drawn off by small mistakes or curving roads.

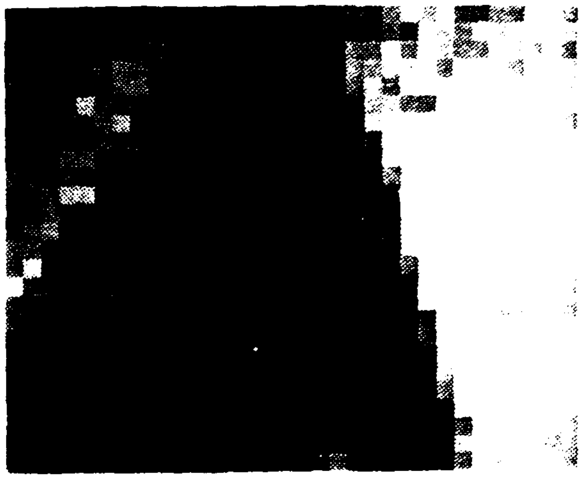


Figure 11: *Low resolution texture image. The brighter blocks are image areas with more visual texture.*

There is no need for complete geometric calibration. The vision algorithms calculate the road's shape (road width and location of the horizon) from the first training image. We also take two calibration pictures, with a meter stick placed perpendicular to the vehicle, 8 and 12 meters in front. Then, during the run, given the centerline of a detected road in image coordinates, it is easy to get the x position of the road at 8 and 12 meters, and then to calculate the vehicle's position on the road.

This algorithm runs in about 10 seconds per image on a dedicated Sun 3/160, using 480 by 512 pixel images reduced to 30 rows by 32 columns. We currently process a new image every 4 meters, which gives about three fourths of an image overlap between images. Ten seconds is fast enough to balance the rest of the system, but is slow enough that clouds can come and go and lighting conditions change between images. We plan to port this algorithm to the Warp, CMU's experimental high-speed processor. We hope to process an image per second, and to use higher resolution.

4. 3D Perception

Our obstacle detection starts with direct range perception using an ERIM scanning laser rangefinder. *Our ERIM produces, every half second, an image containing* 64 rows by 256 columns of range values (see figure 12). The scanner measures the phase difference between an amplitude-modulated laser and its reflection from a target object, which in turn provides the distance between the target object and the scanner. The scanner produces a dense range image by using two deflecting mirrors, one for the horizontal scan lines and one for vertical motion between scans. The volume scanned is 80 degrees wide and 30 high. The range at each pixel is discretized over 256 levels from zero to 64 feet.

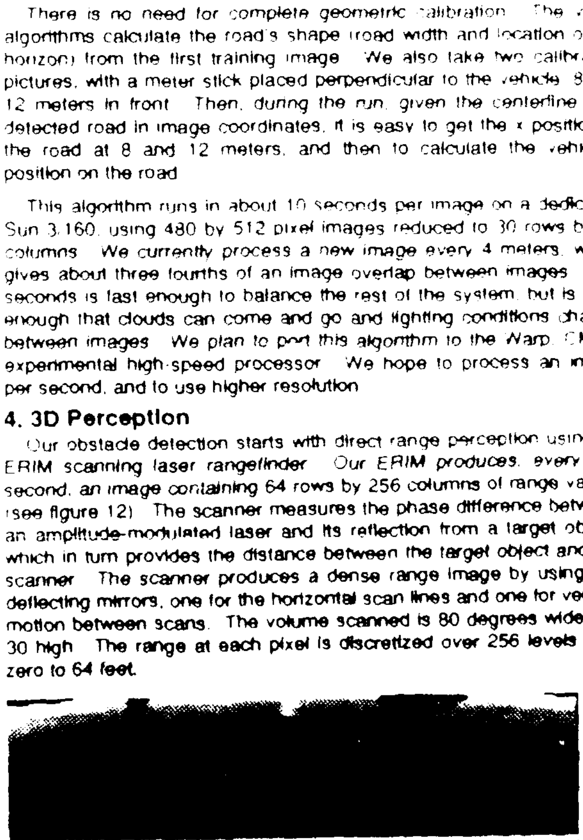


Figure 12: Range image of two trees on flat terrain. Gray levels encode distance, nearer points are painted darker.

Our range processing begins by smoothing the data and undoing the peculiarities of the ranging geometry. The "ambiguity intervals", where range values wrap around from 255 to 0, are detected and unfolded. Two other undesirable effects are removed by the same algorithm. The first one is the presence of mixed points at the edge of an object. The second is that a measurement from a surface such as water, glass, or glossy pigments is meaningless. In both cases, the resulting points are in regions limited by considerable jumps in range and can therefore be removed by the same algorithm. We then transform the values from angle-angle-range, in scanner coordinates, to x-y-z locations. These 3D points are the basis for all further processing.

We have two main processing modes: obstacle detection and terrain analysis. Obstacle detection starts by calculating surface normals from the x-y-z points. Flat, traversable surfaces will have vertical surface normals. Obstacles will have surface patches with normals pointed in other directions. This analysis is relatively fast, running in about 5 seconds on a Sun 3/75, and is adequate for smooth terrain with discrete obstacles.

Simple obstacle maps are not sufficient for detailed analysis. For greater accuracy we do more careful terrain analysis and combine sequences of images corresponding to overlapping parts of the environment into an *extended obstacle map*. The terrain analysis algorithm first attempts to find groups of points that belong to the same surface, and then uses these groups as seeds for the region growing phase. Each group is expanded into a smooth connected surface patch. The smoothness of a patch is evaluated by fitting a surface, plane or quadric. In addition, surface discontinuities are used to limit the region growing phase. The complete algorithm is:

1. Edges: Extract surface discontinuities, pixels with high jumps in x-y-z.
2. Clustering: Find clusters in the space of surface normals and identify the corresponding regions in the original image.
3. Region growing: Expand each region until the fitting error is larger than a given threshold. The expansion proceeds by iteratively adding the point of the region boundary that adds the minimum fitting error.

The clustering step is designed so that other attributes such as color or curvature can also be used to find potential regions on the object. The primitive surface used to compute the fitting error can be either a plane or a quadric surface. The decision is based on the size of the region.

Obstacle detection works at longer range than terrain analysis. When the scanner is looking at distant objects, it has a very shallow depression angle. Adjacent scanlines, separated by 1/2 degree in the range image, can strike the ground at widely different points. And since the grazing angle is shallow, very little of the emitted laser energy returns to the sensor, producing noisy pixels. Noisy range values, widely spaced, make it difficult to do detailed analysis of flat terrain. But a vertical obstacle, such as a tree, shows up much better in the range data. Pixels from neighboring scanlines fall more closely together, and with a more nearly perpendicular surface the returned signal is stronger and the data is cleaner. So it is much easier for obstacle detection to find obstacles than for terrain analysis to certify that a patch of ground is smooth and level.

There are cases in which neither video nor range alone provide enough information, where we have to do data fusion to determine mobility or recognize an object. One such example occurs in navigating the smaller Terregator vehicle around campus sidewalks. At one spot, a sidewalk goes up a flight of stairs and a bicycle path curves around. Video alone has a tough time distinguishing between the cement stairs and the cement bicycle path. Range data cannot tell the difference between the smooth rise of the grassy hill and the smooth bicycle ramp. The only way to correctly identify the safe vehicle path is to use both kinds of data.

We start by fusing the data at the pixel level. For each range point, we find the corresponding pixel in the video image. We produce a painted range image, in which each pixel is a (red, green, blue, x, y, z) 6-vector. Then we can run our standard range segmentation and color segmentation programs, producing regions of smooth range or constant color. For the stairs in particular, we have a special-purpose step detection program that knows about vertical and horizontal planes, and how they are related in typical stairs. It is easy to combine the regions from these separate processes, since they are all in the same coordinates of the painted range image. The final result is a smooth concrete region, in which it is safe to drive, and a positive identification and 3D location of the stairs, for updating the vehicle position.

5. System Building

Besides the problems in building the various modules for road finding, obstacle detection, path planning, and so forth, there are also a set of issues related to building a coherent system. We have built a layered system, with CODGER providing tools and services, and an architecture on top setting conventions for control and data flow.

5.1 Blackboards and Whiteboards

The program organization of the NAVLAB software is shown in figure 13. Each of the major boxes represents a separately running program. The central database, called the *Local Map*, is managed by a program known as the Local Map Builder *(LMB)*. Each module stores and retrieves information in the database through a set of subroutines called the *LMB Interface* which handle all communication and synchronization with the LMB. If a module resides on a different processor than the LMB, the LMB and LMB Interface will transparently handle the network communication. The Local Map, LMB, and LMB Interface together comprise the CODGER (COmmunications Database with GEometric Reasoning) system.

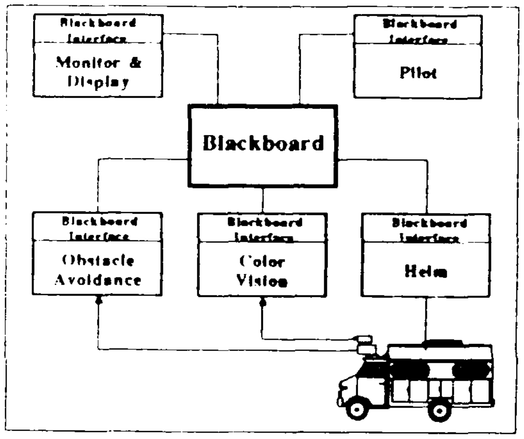


Figure 13: Navlab software architecture

The overall system structure—a central database, a pool of knowledge-intensive modules, and a database manager that synchronizes the modules—is characteristic of a traditional blackboard system. Such a system is called "heterarchical" because the knowledge is scattered among a set of modules that have access to data at all levels of the database (i.e., low-level perceptual processing ranging up to high-level mission plans) and may post their findings on any level of the database; in general, heterarchical systems impose *de facto* structuring of the information flow among the modules of the system. In a traditional blackboard, there is a single flow of control managed by the database (or blackboard) manager. The modules are subroutines, each with a pre-determined precondition (pattern of data) that must be satisfied before that module can be executed. The manager keeps a list of which modules are ready to execute, and in its central loop it selects one module, executes it, and adds to its ready-list any new modules whose preconditions become satisfied by the currently executing module. The system is thus synchronous and the manager's function is to focus the attention of the system by selecting the "best" module from the ready-list on each cycle.

We call CODGER a *whiteboard* because it also implements a heterarchical system structure, but differs from a blackboard in several key respects. In CODGER, each module is a separate, continuously running program; the modules communicate by storing and retrieving data in the central database. Synchronization is achieved by primitives in the data retrieval facilities that allow, for example, for a module to request data and suspend execution until the specified data appears. When some other module stores the desired data, the first module will be re-activated and the data will be sent to it. With CODGER a module programmer thus has control over the flow of execution within his module and may implement real-time loops, demons, data flows among cooperating modules, etc. CODGER also has no pre-compiled list of data retrieval specifications: each time a module requests data, it provides a pattern for the data desired at that time. We call such a system a whiteboard—it is heterarchical like a blackboard, but each module runs in parallel with the module programmer controlling the synchronization and data retrieval requests as best suited for each module. Like other recent distributed AI architectures, whiteboards are suited to execution on multiple processors.

5.2 Data Storage and Retrieval

Data in the CODGER database (Local Map) is represented in *tokens* consisting of classical *attribute-value pairs*. The types of tokens are described in a *template file* that tells the name and type of each attribute in tokens of each type. The attributes themselves may be the usual scalars (integers, floating-point values, strings, enumerated types), arrays (or sets) of these types (including arrays of arrays), or geometric locations as described below. CODGER automatically maintains certain attributes for each token: the token type and id number, the "generation number" as the token is modified, the time at which the token was created and inserted into the database, and the time at which the sensor data was acquired that led to the creation of this token. The LMB Interface provides facilities for building and dissecting tokens and attributes within a module. Rapid execution is supported by mapping the module programmer's names for tokens and attributes onto globally used index values at system startup time.

A module can store a token by calling a subroutine to send it to the LMB. Tokens can be retrieved by constructing a pattern called a *specification* and calling a routine to request that the LMB send back tokens matching that specification. The specification is simply a boolean expression in which the attributes of each token may be substituted; if a token's attributes satisfy the boolean expression, then the token is sent to the module that made the request. For example, a module may specify:

tokens with **type** *equal to "intersection" and* **traffic-control** *equal to "stop-sign"*

This would retrieve all tokens whose **type** and **traffic-control** attributes satisfy the above conditions. The specification may include computations such as mathematical expressions, finding the minimum value within an array attribute, comparisons among attributes, etc. CODGER thus implements a general database. The module programmer constructs a specification with a set of subroutines in the CODGER system.

One of the key features of CODGER is the ability to manipulate geometric information. One of the attribute types provided by CODGER is the *location*, which is a 2-D- or 3-D polygon and a reference to a *coordinate frame* in which that polygon is described. Every token has a specific attribute that tells the location of that object in the Local Map, if applicable, and a specification can include geometric calculations and expressions. For example, a specification might be:

tokens with **location** *within 5 units of (45,32) [in world coordinates]*

or

tokens with **location** *overlapping X*

where *X* is a description of a rectangle on the ground in front of the vehicle. The geometric primitives currently provided by CODGER include calculation of centroid, area, diameter, convex hull, orientation, and minimum bounding rectangle of a location, and distance and intersection calculations between a pair of locations. We believe that this kind of geometric data retrieval capability is essential for supporting spatial reasoning in mobile robots with multiple sensors. We expect geometric specifications to be the most common type of data retrieval request used in the NAVLAB.

CODGER also provides for automatic coordinate system maintenance and transformation for these geometric operations. In the Local Map, all coordinates of location attributes are defined relative to **WORLD** or **VEHICLE** coordinates; **VEHICLE** coordinates are parameterized by time, and the LMB maintains a time-varying transformation between **WORLD** and **VEHICLE** coordinates. Whenever new information (i.e., a new **VEHICLE**-to-**WORLD** transform) becomes available, it is added to the "history" maintained in the LMB; the LMB will interpolate to provide intermediate transformations as needed. In addition to these basic coordinate systems, the LMB Interface allows a module programmer to define *local coordinates* relative to the basic coordinates or relative to some other local coordinates. Location attributes defined in a local coordinate system are automatically converted to the appropriate basic coordinate system when a token is stored in the database. CODGER provides the module programmer with a conversion routine to convert any location to any specified coordinate system.

All of the above facilities need to work together to support asynchronous sensor fusion. For example, suppose we have a vision module A and a rangefinder module B whose results are to be merged by some module C. The following sequence of actions might occur:

1. A receives an image at time 10 and posts results on the database at time 15. Although the calculations were carried out in the camera coordinate system for time 10, the results are automatically converted to the **VEHICLE** system at time 10 when the token is stored in the database.

2. Meanwhile, B receives data at time 12 and posts results at time 17 in a similar way.

3. At time 18, C receives A's and B's results. As described above, each such token will be tagged with the time at which the sensor data was gathered. C decides to use the vehicle coordinate system at time 12 (B's time) for merging the data.

4. C requests that A's result, which was stored in **VEHICLE** time 10 coordinates, be transformed into **VEHICLE** time 12 coordinates. If necessary, the LMB will automatically interpolate coordinate transformation data to accomplish this. C can now merge A's and B's results since they are in the same coordinate system. At time 23, C stores results in the database, with an indication that they are stored in the coordinate system of time 12.

5.3 Synchronization Primitives

CODGER provides module synchronization through options specified for each data retrieval request. Every time a module sends a specification to the LMB to retrieve tokens, it also specifies options that tell how the LMB should respond with the matching tokens:

- *Immediate Request.* The module requests all tokens currently in the database that match this specification. The module will block (i.e., the "request" subroutine in the LMB Interface will not return control) until the LMB has responded. If there are no tokens that match the specification, the action taken is determined by an option in the module's request:
  - *Non-Blocking.* The LMB will answer that there are no matching tokens, and the module can then proceed. This would be used for time-critical modules such as vehicle control. Example: "Is there a stop sign?"
  - *Blocking.* The LMB will record this specification and compare it against all incoming tokens. When a new token matches the specification, it will be sent to the module and the request will be satisfied. Meanwhile, the module will remain blocked until the LMB has responded with a token. This is the type of request used for setting up synchronized sets of communicating modules: each one waits for the results from the previous module to be posted to the database. Example: "Wake me up when you see a stop sign."
- *Standing Request.* The module gives a specification along with the name of a subroutine. The module then continues running; the LMB will record the specification and compare it with all incoming tokens. Whenever a token matches, it will be sent to the module. The LMB Interface will intercept the token and execute the specified subroutine, passing the token as an argument. This has the effect of invoking the given subroutine whenever a token appears in the database that matches the given specification. It can be used at system startup time for a module programmer to set up "demon" routines within the module. Example: "Execute that routine whenever you see a stop sign."

5.4 Architecture

There are several modules that use the CODGER tools, and that fit into a higher level architecture. The modules are:

- Pilot. Looks at the map and at current vehicle position to predict road location for Vision. Plans paths.
- Color Vision. Waits for a prediction from the Pilot, waits until the vehicle is in the best position to take an image of that section of the road, returns road location.
- Obstacle Avoidance. Gets a request from the Pilot to check a part of the road for obstacles. Returns a list of obstacles on or near that chunk of the road.
- Helm. Gets planned path from Pilot, converts polyline path into smooth arcs, steers vehicle.
- Graphics and Monitor. Draws or prints position of vehicle, obstacles, predicted and perceived road.

There are three other modules in our architecture but not yet implemented

- Captain. Talks to the user and provides high-level route and mission constraints such as "avoid area A" or "go by road B"
- Map Navigator. Maintains a map, does global path planning, provides long-term direction to the Pilot.
- Lookout. Looks for landmarks and objects of importance to the mission.

These modules use CODGER to pass information about "driving units". A driving unit is a short chunk of the road or terrain (in our case 4 meters long) treated as a unit for perception and path planning. The Pilot gives driving unit predictions to Color Vision, which returns an updated driving unit location. Obstacle Detection then sweeps a driving unit for obstacles. The Pilot takes the driving unit and obstacles, plans a path, and hands the path off to the Helm. The whole process is set up as a pipeline, in which Color Vision is looking ahead 3 driving units, Obstacle Detection is looking 2 driving units ahead, and path planning at the next unit. If for any reason some stage slows down, all following stages of the pipeline have to wait. So, for instance, if Color Vision is waiting for the vehicle to come around a bend so it can see down the road, Obstacle Detection will finish its current unit and will then have to wait for Color Vision to proceed. In an extreme case, the vehicle may have to come to a halt until everything clears up. All planned paths include a deceleration to a stop at the end, so if no new path comes along to overwrite the current path the vehicle will stop before driving into an area that has not been seen or cleared of obstacles.

In our current system and test area, three driving units is too far ahead for Color Vision to look, so both Color Vision and Obstacle Detection are looking at the same driving unit. Obstacle Detection looks at an area enough larger than the Pilot's predicted driving unit location that the actual road is guaranteed to be covered. Another practical modification is to have Obstacle Detection look at the closest driving unit also, so if a person walks onto the road immediately in front of the vehicle he will be noticed. Our system will try to plan a path around obstacles while remaining on the road. If that is not possible, it will come to a halt and wait for the obstacle to move before continuing.

6. Conclusions and Future Work

The system described here works. Since quashing the last of our (known) bugs, it has successfully driven the Navlab many tens of times, processing thousands of color and range images without running off the road or hitting any obstacles. CODGER has proved to be a useful tool, handling a lot of the details of communications and geometry. Module developers have been able to build and test their routines in isolation, with relatively little integration overhead. Yet there are several areas that need much more work.

Speed. We drive the Navlab at 10 cm/sec, a slow shuffle. Part of the reason for this is because our test road is narrow and winding, and part of the reason is that we deliberately concentrate on competence rather than speed. But faster motion is always more interesting, so we are pursuing several ways of increasing speed. One bottleneck is the computing hardware. We are mounting a Warp, CMU's experimental high speed processor, on the Navlab. The Warp will give us a factor of 100 more processing power than a Sun for color and range image processing. At the same time, we are looking at improvements to the software architecture. We need a more sophisticated path planner, and we need to process images that are more closely spaced than the length of a driving unit. Also, as the vehicle moves more quickly, our simplifying assumption that steering is instantaneous and that the vehicle moves along circular arcs becomes more seriously flawed. We are looking at other kinds of smooth arcs, such as clothoids.

Map. One particular reason for the slow speed is that the Pilot assumes straight roads. We need to have a description that allows for curved roads, with some constraints on maximum curvature. The next steps will include building maps as we go, so that subsequent runs over the same course can be faster and easier.

Cross country travel. Travel on roads is only half the challenge. The Navlab should be able to leave roads and venture cross country. Our plans call for a fully integrated on-road/off-road capability.

Intersections. Current vision routines have a built in assumption that there is one road in the scene. When the Navlab comes to a fork in the road, vision will report one or the other of the forks as the true road depending on which looks bigger. It will be important to extend the vision geometry to handle intersections as well as straight roads. We already have this ability on our sidewalk system, and will bring that over to the Navlab.

Landmarks. Especially as we venture off roads, it will become increasingly important to be able to update our position based on sighting landmarks. This involves map and perception enhancements, plus understanding how to share limited resources, such as the camera, between path finding and landmark searches.

Software Development. Our current blackboard system can manipulate primitive data elements but has no concept of data structures made up of tokens on the blackboard. We need aggregate data types for representing complex 3D geometric descriptions of objects for recognition. We will also be implementing a LISP interface to our blackboard so that not all modules need to be written in C.

Integration with Work from Other Sites. There are other universities and research groups cooperating with Carnegie Mellon through DARPA's Strategic Computing Vision program. We plan to incorporate some of their programs into the Navlab system in the coming years as it evolves into the "New Generation Vision System" which is that goal of that program.

Acknowledgements

This research is supported in part by the Advanced Research Projects Administration of the Department of Defense (DARPA), under contract DACA 76 85-C-003, monitored by the US Army Engineer Topographic Laboratories. We are part of the Strategic Computing Vision effort.

The Terregator and Navlab were built by William Whittaker's group in the Construction Robotics Laboratory, and the Warp group is led by H. T. Kung. Faculty in the ALV group include Martial Hebert (range finder work) and Jon Webb (Warp vision). The real work gets done by an army of 8 staff, 9 graduate students, 5 visitors, and 3 part time programmers.

References

1. J. Crisman. "Machine Perception." *Unix Review 4*, 9 (1986).
2. Elfes, A. A Sonar-Based Mapping and Navigation System. IEEE International Conference on Robotics and Automation, 1986.
3. Y.Goto, K.Matsuzaki, I.Kweon, T.Obatake. CMU Sidewalk Navigation System. Fall Joint Computer Conference, ACM/IEEE, November, 1986.
4. Hebert, M., and Kanade, T. Outdoor Scene Analysis Using Range Data. IEEE International Conference on Robotics and Automation, 1986.
5. Kanade, T., Thorpe, C., and Whittaker, W. Autonomous Land Vehicle Project at CMU. ACM Computer Conference, Feb, 1986.
6. Takeo Kanade and Charles Thorpe. CMU Strategic Computing Vision Project Report: 1984 to 1985. The Robotics Institute, Carnegie-Mellon University, 1985.
7. Krogh, B., and Thorpe, C. Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles. IEEE International Conference on Robotics and Automation, 1986.

8. L.H. Matthies and S.A. Shafer. Error modelling in stereo navigation. Fall Joint Computer Conference, ACM/IEEE, November, 1986.
9. B. Serey and L. Matthies. Obstacle avoidance using 1-D stereo vision. Carnegie Mellon Robotics Institue, 1987.
10. Shafer, S., Stentz, A., Thorpe, C. An Architecture for Sensor Fusion in a Mobile Robot. IEEE International Conference on Robotics and Automation, 1986.
11. *Jeff Singh et al. NavLab: An Autonomous Vehicle.* Carnegie Mellon Robotics Institute, 1986.
12. C. Thorpe. Vision and Navigation for the CMU Navlab. SPIE, Society of Photo-Optical Instrumentation Engineers, October, 1986.
13. Wallace, R., Matsuzaki, K., Goto, Y., Crisman, J., Webb, J., and Kanade, T. Progress in Robot Road-Following. IEEE International Conference on Robotics and Automation, 1986.

[illegible] 1606

AIAA-87-1667
The Connection Machine: A Fine Grained Multi-Processor

W. D. Hillis, Thinking Machines Corp.,Cambridge, MA

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

March 9-11, 1987/Arlington, VA

THE CONNECTION MACHINE: A FINE GRAINED MULTI-PROCESSOR

W. Daniel Hillis
Founding Scientist
Thinking Machines Corporation
245 First Street
Cambridge, MA 02142

Abstract

The current upper limit on the capacity of computing systems is determined by our ability to design computers, not by demand. Advances in circuit fabrication technology have recently made possible the construction of a new type of computing engine called the Connection Machine which contains tens of thousands of tiny processors. Each processor is capable of computing concurrently, so the machine is much faster than a conventional computer.

Background

The first prototype of the Connection Machine system, with 65,536 processors, was constructed by Thinking Machines Corporation under contract from DARPA in 1985. Even this first prototype, the CM1, is one of the world's fastest computers, and it is currently being applied to problems in fluid dynamics, database retrieval, design automation, image processing, simulation, and artificial intelligence. A commercial version of the CM1 was introduced in spring of 1986, and is currently being sold and manufactured by Thinking Machines Corporation.

The key idea of the Connection Machine is to combine memory and processing into a single cell which can be replicated many times. Most computers today are based on a two-part design where the memory and processing functions are performed by separate units. This two-part design made a great deal of sense in the days when processing was realized by relatively expensive and fast vacuum tubes and memory was realized by relatively slow and inexpensive delay lines. Since the two functions of the computer were implemented by essentially different technologies, they were implemented separately. The system was programmed in such a way as to keep the expensive vacuum tubes relatively busy. Programmers were trained to break up a problem into a series of sequential operations, so that the data could be processed one word at a time. For example, to simulate that flow of a fluid on a conventional machine the volume of the velocities of various points in space would be stored in the memory and cut up into cells. To simulate the change in fluid over a moment of time, the programmer must arrange to sequentially shift the cells to the processing unit one at a time for computation. Although the computation would be modelling a system that is fundamentally parallel, namely the flow of the fluid, the programmer would realize this by a series of sequential operations.

This sequential method of solving problems made a great deal of sense in the 1940's, since it allowed only one processing unit for a large amount of memory. The economics of the technology at the time were such that the single processing unit was as large and expensive as many thousands of words of memory. Today we still use word-at-a-time processing in almost all our computers even though the technical reasons for using it have largely vanished. Both memory and processing elements today are built of the same fundamental material; usually silicon. They can be fabricated on the same integrated circuit. In spite of this, a modern single chip computer is essentially just a scaled-down version of the roomful of vacuum tubes and mercury delay lines that represented a computer in the 1950's. The organization of programs in terms of streaming memory past a single processor, although originally developed for reasons of efficiency, has become a serious performance handicap. Today most of the silicon in a computer is in the memory, and most of this is idle.

One way to speed up a computer is to increase the speed of the primitive operations. This is difficult to do beyond a certain point. The fastest conventional machines already have operation times comparable to the time required for light to travel from one side of the processor to the other. While some improvement can be made by making the processing components physically smaller, the operation times of processors are getting close to their practical limits. It is becoming increasingly difficult to improve the performance of computers without fundamentally changing the design. Since such speed improvements require the use of faster components and denser packaging, large computers today are significantly less economical than their smaller scale counterparts in terms of cost per computation.

The solution is to process the data in parallel. We can take advantage of the fact that memory and processing components are fabricated on the same silicon chip and produce a simple processor memory unit that can be replicated tens of thousands of times. We can also take advantage of a more natural mapping of the problem into the machine by eliminating the need to arrange the computations in serial order.

The construction of a machine with hundreds of thousands of processors is made possible by the availability of very large scale integrated circuits with hundreds of thousands of transistors on a single chip. But just having many processors is not sufficient. Very few interesting applications decompose into totally independent subproblems; most require communication between the processing elements. For example, in a three-dimensional fluid flow problem each unit of volume must communicate with its immediate neighbors in three-dimensional space. In other problems, such as a large data base problem, the pattern of the communication can be much more complicated, depending entirely upon the content of the data. In fact, many problems require dynamically changing patterns of communication that rearrange during the course of computation.

This leads to the second important element of the Connection Machine architecture: the connections. Each processor/memory cell is linked by a communications network to all other cells in the machine. This arrangement is similar to giving each cell a telephone with which it can call up any other cell with which it needs to communicate. The packet switch communication network serves the role of the telephone system. Cells can communicate whenever they know one another's "telephone numbers." In a three-dimensional problem, such as modelling the flow of fluid over an airplane wing, each cell is given the numbers of the cell representing its three-dimensional neighbors. Cells do not have a physical three-dimensional pattern of wiring, but many applications involve a more complex pattern of connections. For an example of an application that requires a more complex pattern of communications, consider the simulation of an electronic circuit with a hundred thousand transistors. Such simulations are common and commercially important during the design of integrated circuits. In such an application it is natural to use one processing/memory unit to represent the state of each of the hundred thousand transistors. The natural pattern of communication depends on the exact wiring pattern of the transistors. If two processor cells simulate two transistors which are connected in the diagram, then the processor cells must communicate. When the voltage level on one transistor changes, this change is propagated to the other transistor through the connection. In such an application the processing cells are assigned to transistors arbitrarily, and each processing cell is given the telephone numbers of the transistors to which it is connected. Thus the communications structure of the processing elements exactly matches the natural structure of the problem.

In more complex algorithms, such as those that occur in the study of artificial intelligence, the connection pattern may actually change during the course of a computation. For example, each processor may represent a simple concept, with the connections between the processors representing the relationships between the concepts. One processor may represent the concept "Socrates," another "man," and another "mortal." A connection between the processor representing "Socrates" and the processor representing "man" represents the concept that "Socrates is a man." And, similarly, a connection between man and mortal may represent the statement that "all men are mortal." Such a machine would be able to deduce the concept that Socrates was mortal by following the chain of connections. The example is simple and contrived. But in a real artificial intelligence application, a simple common-sense knowledge base might consist of hundreds of thousands of such assertions; for example, assertions like "dogs are animals" and "apples are often red." From such a knowlege base a program may be required to make hundreds of common-sense inferences per second. Each time a new fact is learned, a new connection must be formed.

The implementation of the communication network is the most difficult technical problem in constructing a Connection Machine system. For example, in the first 65,536 of the machine, new connections are formed at a rate of about one hundred million each second. This is comparable to the switching capacity of the entire telephone exchange for the city of New York.

The key component that makes this all work is an integrated circuit that integrates 16 simple processing units and a communications router. Each of the processinc units is a very simple computer with 4,000 bits of memory. The difficult part is the router. There are 4,096 chips in the 65,536 processor system. Packaging considerations make it impractical to have a direct connection between all pairs of chips. Instead, each router is connected directly to only 12 others in the system. Two processors within a single chip may communicate directly, but processors on different chips may need to communicate through intermediaries. This is the function of a router. It is the responsibility of a router to forward an incoming message from one chip to another. The pattern of wiring is such that six such forwarding steps, on the average, are required to communicate between two chips. Even with all the forwarding steps, the transmission of a message from one processor to another requires less than 100 microseconds. Each packet is transmitted as a stream of bits that are forwarded through the system in such a way that the beginning of the stream is usually delivered before the end is even transmitted.

The entire 65,536 processor CM1, including processor/memory units and the communication networks, fits into a cube approximately five feet on a side. It is controlled by a conventional host computer so that the operating system and user interface seen by the user are those of the host. In fact, the Connection Machine connects to the host computer in much the same way as a conventional memory unit. The processor/memory cells can be accessed simply as memory cells by the host. This arrangement allows simple integration of parallel computing and existing software, since data structures may be shared by both the Connection Machine and the conventional von Neumann host.

The design of the 65,536 processor machine was optimized for simplicity and reliability rather than speed. It runs at a relatively slow clock rate of 250 nanoseconds, as compared to, say, 25 nanoseconds for a typical supercomputer. There are no exotic technologies to the machine. The custom chip is built with a simple integrated circuit process similar to that used in personal computers and pocket calculators. The circuit board's cooling, packaging, and power systems are all similar to those found in conventional computers. Even with this conservaive technology, the Connection Machine is able to achieve computing rates of greater than 1000 million instructions per second, more than 100 times the speed of a typical large computer. The instruction set of the machine is optimized for operating on problems that involve images and words rather than just numbers. For this type of problem the machine is able to achieve peak rates of over 50,000 million instructions per second.

AIAA-87-1674 Computational Themes in Applications of Visual Perception

R. Jain, B. G. Schunck and T. Weymouth,
University of Michigan,
Ann Arbor, MI

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program
March 9-11, 1987/Arlington, VA

Computational Themes in Applications of Visual Perception

Ramesh Jain
Brian G. Schunck
Terry Weymouth

Computer Vision Research Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122

abstract

The paper summarizes the current research in the Computer Vision Research Laboratory at the University of Michigan. The laboratory concentrates on developing generic vision algorithms for industrial applications. Generic vision algorithms can be applied to a wide variety of inspection problems. The paper includes a discussion of the current state of the machine vision industry and provides recommendations for improving the transfer of vision technology from research to practice.

1 Introduction

The University of Michigan recently formed a laboratory, called the Computer Vision Research Laboratory, within the Department of Electrical Engineering and Computer Science for research in computer vision. The Computer Vision Research Laboratory will evolve through interaction among the researchers of various related disciplines and through inductrial and government contacts.

The research program in the Computer Vision Laboratory concentrates on the development of generic vision algorithms that can be applied to a variety of situations. Generic perception algorithms are simple, reliable modules that hide the details of perception in a packaged hardware or software component that can be used by applications specialists who are not themselves expert vision researchers. Generic vision algorithms are not developed for specific applications, but are designed to solve a vision task that is part of many different inspection problems that occur in different applications in different industries. In developing generic vision algorithms, researchers leverage their efforts by providing solutions to classes of vision tasks while removing the details of the application from the research efforts. This insulates the vision researchers from the details of the application and allows them to concentrate on understanding the vision problems.

Another aspect of research on generic vision algorithms is that the algorithms are classified according to properties that are meaningful in the context of the emerging knowledge of vision fundamentals, as opposed to classification according to applications areas. There are no automotive vision algorithms or aerospace vision algorithms; vision algorithms are not specific to any one industry. Vision algorithms are defined by the properties of the vision processing task; that is, in terms that depend on the vision processing itself, rather than on some intended use for the algorithm.

The nature of the problems of applying vision to specific applications has not been widely perceived. The current macroeconomic system of machine vision is inappropriate. The solution is to place machine vision technology in the hands of the end users who are familiar with the requirements of their environment. Vision technology should be like any other instrumentation: plug it in and use it to make measurements. It is not practical for the developers of machine vision applications to perform the development specific to the application and to develop the "instrumentation" required for the development at the same time. To change the current, untenable system, vision researchers must produce generic vision algorithms that can be used by vision systems developers in a variety of applications.

Researchers are currently addressing problems related to vision engineering but without much, if any, concern for designing machine vision systems. The focus of the Computer Vision Research Laboratory on vision algorithms that are generic, rather than applications specific, meets needs not addressed by other research institutions. The Computer Vision Research Laboratory provides a focus for the synergistic interaction of researchers aided by interactions with industry. The goal of research in the laboratory is to address fundamental problems blocking advances in the design of machine vision systems for various applications. The research addresses basic issues in computational vision research and the design of systems that can perform vision tasks in real time for a given application. New engineering techniques for designing systems are being developed that can perform tasks in an unstructured environment. The research may also advance the understanding of biological vision systems.

2 Research Problems

Visual perception is the process of interpreting measurements such as light intensity and range that relate to the projection of surfaces in a scene onto the image plane. The central problem in vision is the reconstruction of surface structure and properties from the projection onto the image plane. This paper outlines some of the research problems currently being explored in the Computer Vision Research Laboratory.

- Dynamic vision
- Range images
- Knowledge-based perception svstems
- Computer architectures for vision algorithms
- Sensor integration
- Bridging the gap between research results and applications

3 Dynamic Vision

Motion provides valuable clues to surface structure. Algorithms for interpreting sequences of images can provide measurements that are very useful for applications. Image flow research has many applications including passive sensing systems for autonomous mobility, machine inspection of surface structure, passive sensors for aircraft and satellite docking systems, and image compression.

3.1 Background

The last few years have seen increasing interest in dynamic scene analysis. The input to a dynamic scene analysis system is a sequence of images. A major problem in a computer vision system is to recover the information about objects in a scene from images. This problem cannot be solved without some assumptions about the world. A sequence of frames provides the additional dimension of time for recovering the information about a 3-dimensional world that is lost in the projection process. Multiple views of a moving object acquired using a stationary camera may allow recovery of the structure of the object [45]. A mobile camera may be used to acquire information about the structure of the stationary objects in a scene using optical flow [11], axial motion stereo [25,32], and other methods [21].

Many researchers in the psychology of vision support the recovery of information from image sequences, rather than from a single image [11,31]. Gibson [11] argued in support of active information pick-up by the observer in an environment. Neisser [31] proposed a model in which the perceptual processes continually interact with the incoming information to verify expectations formed on the basis of available information. In computer vision systems, the power of exploiting motion, even with such noise sensitive approaches as difference pictures and accumulated difference pictures, has been demonstrated on complex, real world scenes [20]. Many researchers are addressing the problem of recovering information in dynamic scenes; but due to the legacy of static scenes, most researchers are approaching the recovery problem using just two or three frames of a sequence. This restricts the results to quasi-dynamic scene analysis, rather than true dynamic scene analysis. The information recovery process requires constraints about the scene. The analysis based on small numbers of frames rests on assumptions that ignore the most important information in dynamic scenes.

The viewpoint of the Computer Vision Research Laboratory is that image understanding is a dynamic process. Dynamic vision algorithms cope with the error-filled visual world by exploiting redundant information in the image sequence. Current research efforts are developing a qualitative approach to vision that uses only relative information available in a sequence to infer relationships between objects in a scene.

3.2 Segmentation

In many dynamic scene analysis systems, the goal is to recognize moving objects and to find their motion characteristics. If the scene is acquired using a stationary camera, then segmentation generally refers to the separation of moving components from stationary components in the scene and identification of individual moving objects based on velocity or some other characteristic. For the case of a moving camera, the segmentation task may be the same as above or may involve further segmentation of the stationary components of the scene by exploiting the motion of the camera. Most research efforts for the segmentation of dynamic scenes have been concerned with the extraction of the images of the moving objects observed by a stationary camera. It has long been argued by researchers in perception [11,45] that motion cues aid the segmentation process. Computer vision techniques for segmenting dynamic scenes perform well in comparison to those for the segmentation of stationary scenes. The segmentation of moving camera scenes into their stationary and nonstationary components has received attention only recently [22]. The major problem in the segmentation of moving observer scenes is that every surface in the scene shows motion in the image. For separation of moving object images, the motion component assigned to various surfaces in the images due to the motion of the camera must be removed. The fact that the image motion of a surface depends on its distance from the camera and the surface structure complicates the situation.

Jain developed techniques for the segmentation of dynamic scenes [19,20,22]. These techniques have been used in many applications. Sandia Laboratories is developing systems for tracking objects for Army applications using techniques based on differencing using a likelihood ratio. Recently research led to a new approach for segmentation using accumulative difference pictures that may be implemented easily on special hardware [23].

3.3 Image Flow

Image flow is the velocity field in the image plane that arises due to the projection of moving patterns in the scene onto the image plane. The motion of patterns in the image plane may be due to the motion of the observer, the motion of objects in the scene, or both. The motion may also be apparent motion where a change in the image between frames gives the illusion of motion [36].

The intent of this research is to discover new models for image flow that will yield new algorithms for image flow estimation and analysis. Constraint equations that better model the interactions between changes in object position and the illumination and surface reflectance characteristics will naturally result in better algorithms and better understanding of existing algorithms.

Prior image flow research developed an image flow equation for smooth patterns of image irradiance and smooth velocity fields [27]. The equation was extended first to image irradiance patterns with discontinuities and then to velocity fields with discontinuities [55,56,60]. Future research in image flow constraint equations will aim to increase our understanding of image flow characteristics. This will lead to new algorithms for image flow estimation that incorporate the new continuity equations. Restrictions on the situations in which existing continuity equations can be used will be discovered and these insights will improve the performance of existing image flow estimation algorithms by pin pointing the situations where the algorithms cannot be used.

3.4 Motion Stereo

Depth determination is a continuing problem in computer vision. Research in this area is motivated by the need for target tracking, autonomous vehicles control, visual prostheses for the blind, realistic flight trainers, and models of the human visual system. There is a plethora of dep... determination techniques. Many different stereo systems for depth determination have been developed just in the last few years.

Stereo information can also be obtained using a single moving camera. Jain and O'Brien [25,32] map images into a complex log space where the movement of the objects in two dimensions due to the camera motion becomes a translation along one axis in the new space. Given this phenomenon, the motion correspondence problem is greatly reduced, since only a small strip of the new space needs to be searched [18]. This constraint is similar to the epipolar constraint in stereo. In addition, the transform is scale and rotation invariant. It also has an analog in the human visual system: the mapping of the retinal space into the striate cortex is very closely approximated by the CLM. Figure 1 shows an image and its mapping.

Optical flow has been studied with the aim of recovering information about the environment and the motion of the observer. The *egomotion complex logarithmic mapping* (ECLM) exploits some characteristics of optical flow without computing it. The mapping combines scale, projection, and rotation invariances of complex log mapping with the characteristics of optical flow. This mapping is useful in segmentation of image sequences to recover images of moving objects. The distance to stationary objects can also be computed from the egomotion complex logarithmic mapping. Using this mapping, motion stereo and segmentation can be achieved in one step. The feasibility of the approach has been demonstated. The next step is to study its application in dynamic scenes. This research should yield important results for the navigation of mobile robots.

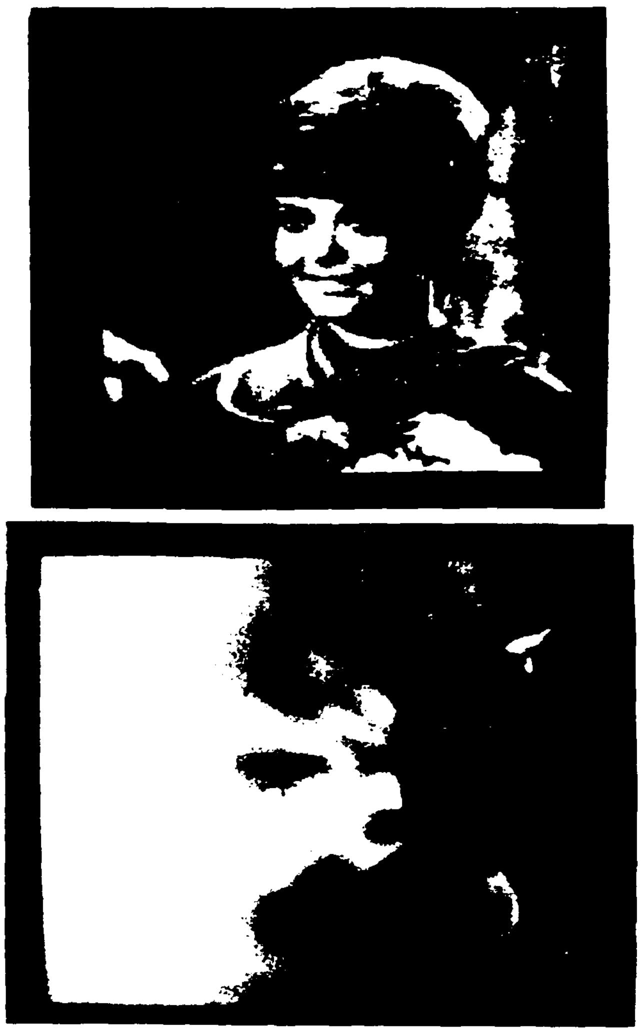


Figure 1: An image and its complex log mapping are shown in this figure. The mapping is similar to the retino-striate mapping in primate visual systems. This mapping is very useful in segmentation and depth recovery in dynamic vision.

3.5 Motion Trajectories

Iterative algorithms for determining trajectories of points in an extended frame sequence using *path coherence* are being developed. The emphasis in this approach is to exploit motion characteristics for establishing correspondence without assuming rigidity of objects. A greedy exchange algorithm has been developed to implement this idea [41]. The results of applying this algorithm to a sequence from the *Superman* movie are shown in Fig. 2.

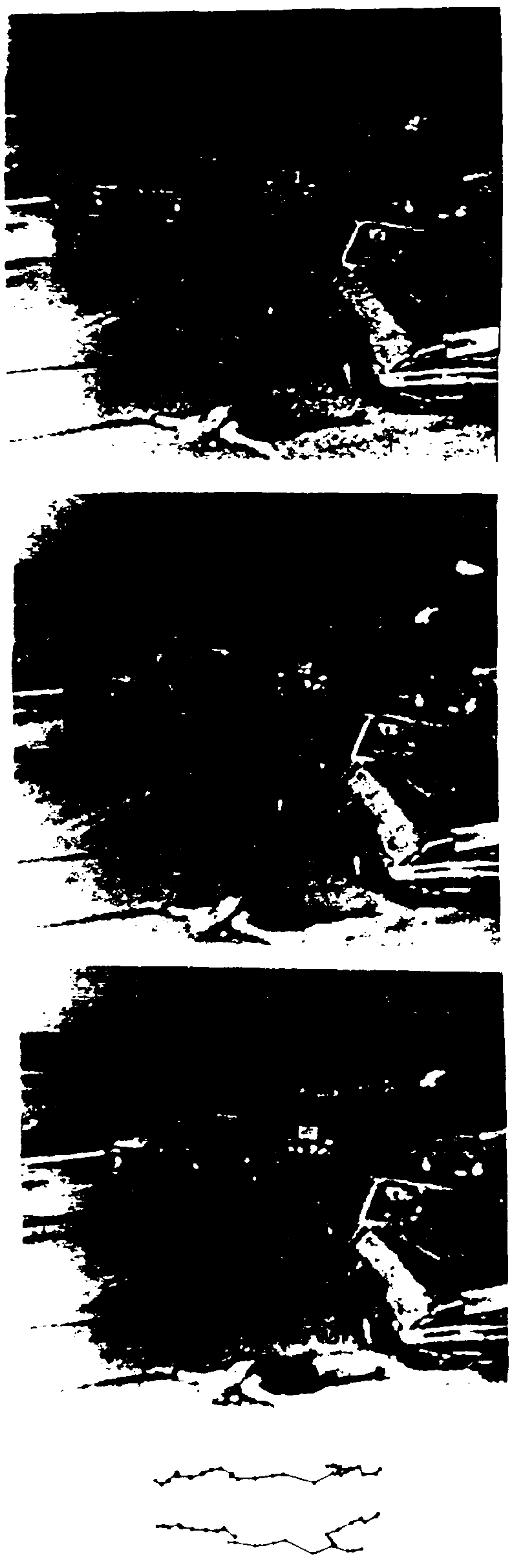


Figure 2: Three frames from the *Superman* sequence are shown in this figure. Points on the head and belt of the three soldiers running towards the camera were tracked using a greedy exchange algorithm. The trajectories are also shown here.

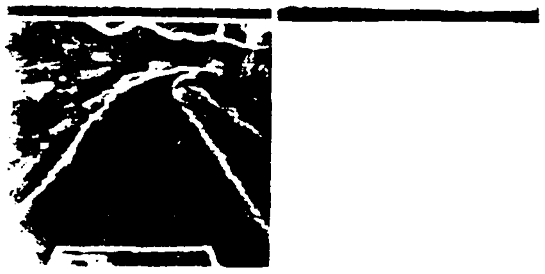


Figure 3: This figure shows results of our road edge detection algorithm. For the road shown in the top left corner, the road edges are shown in the right bottom corner.

Another approach to finding trajectories that will determine motion events is being studied. In this approach, motion parameters are recovered using constraints imposed by the equations for the motion of points [15]. The algorithm uses successive refinement to determine the trajectories of points. A very attractive feature of this approach is that it determines discontinuities and tries to use smoothness only for the known smooth path in establishing correspondence.

The determination and role of events in dynamic scenes is being studied. A motion event may occur due to a change in the motion parameters or due to occlusion. A major thrust of other research is to develop techniques that will recover qualitative information about depth and motion of objects using constraint propagation.

3.6 Navigation

Techniques for the guidance of autonomous mobile robots or vehicles are being developed. This requires techniques to recover information about the environment using vision and then to use knowledge based techniques to control the navigation. The observation that the location of road vanishing points does not change significantly from frame to frame is used to develop an algorithm for finding road boundaries [29]. The algorithm uses the hypothesize and test paradigm. Results of this approach are shown in Fig. 3.

4 Range Images

The long term goal of this project is to develop techniques that will be useful in object recognition and navigation using range information.

The last few years have seen increasing attention to the analysis of range images. Range images may be obtained with passive methods, such as binocular stereo, or with range sensors. Range images contain explicit information about surfaces. This explicit information facilitates recognition and location tasks in many applications. The goal in range image understanding is to find robust symbolic surface descriptions that are independent of viewpoint. Techniques to characterize surfaces in range images are being developed [5,4,24].

Surfaces are segmented using local features, such as Gaussian and mean curvatures and related differential geometric measures. The signs of the curvatures at each point in the image are used to assign one of eight basic surface types to each point in the image. The next step is to develop techniques to identify surfaces by grouping points using the surface type, spatial proximity, and other criteria. All grouping processes must conform to the sensed information since the ultimate truth is in the sensed data. This stimulus bound approach uses symbolic surface descriptions in the segmentation of images. Figure 4 shows a result of this segmentation approach. The symbolic surface descriptions will play important role in many applications.

4.1 Recognition Methods

Object recognition is a major motivation in most image-understanding systems. Despite strenuous efforts, only limited success has been achieved.

The object recognition task can be classified based on difficulty in several ways. One way is based on the degree of uncertainty allowed in the object's position and orientation. This can range from no uncertainty, to uncertainty only in its 2-D position, through uncertainty in both its 2-D position and rotation, up to uncertainty in its 3-D position and orientation.

The task can be further classified based on the complexity of interactions allowed between objects. In the simplest case, each object to be recognized must be completely visible and surrounded by background. In a more complex case, objects are allowed to touch but not overlap. In the most general case, objects are allowed to partially occlude one another.

Most work on the occluded-objects problem has concentrated on the case where there is only one object type in the scene. Algorithms developed for this problem can be extended to cases of multiple objects types, simply by running the algorithm multiple times, once for each possible object type. This is not ideal, as it does not consider or take advantage of the similarities and differences among possible objects. Also the recognition time increases linearly with the number of possible objects, after a fixed time for preprocessing. This can become a problem as the number of possible object types increases.

A new method for object recognition called the *feature indexed hypotheses method* is being developed. This method breaks the recognition process into two phases: hypotheses generation and hypothesis verification. By using features that occur multiple times in the possible object set, the number of features in the search can be greatly reduced generating a small number of false hypotheses that are easily rejected. By carefully selecting the features, the recognition time growth rate can be reduced to the square root of the number of possible objects. This method also has the advantage that unique features which are difficult or impossible to find if the possible object set contains many similar objects are not required. The method's feasibility is demonstrated by the results of a prototype two-dimensional occluded-parts recognition system.

5 Knowledge-Based Perception Systems

Most complex tasks require specialized knowledge, image understanding is no exception. The last few years have seen an increasing application of knowledge in computer vision systems. The application of knowledge at different levels in a vision system is being studied.

5.1 Knowledge-Based Algorithms

An image interpretation system is a program that derives a scene description from a time-varying image. The input signal is a rectangular grid of signal samples taken at regular time intervals $I(x, y, t)$. The sample can be taken in several frequency ranges (for example, red, green, and blue) and combined into a time-varying vector field $C(x, y, t)$. There is wide agreement that this characterizes the environmental data used by such a system [9,17]. On the other hand, it is much more difficult to formulate a description of the output of an interpretation system. In this project, the output of an interpretation system is a network representing the scene being viewed, an approach common to other interpretation systems. The nodes of such a network are groups of objects within the scene: individual objects, object parts, or references to clusters of image events. Arcs in the network are labeled with relations between the objects. Since primitive object parts are usually depicted as geometric solids or collections of joined surface patches [10], these representations are included in the representation used in this work.

However, there is no need to limit ourselves to only that type of representation. In fact, the study of which types of primitives are useful and how they relate to the processing of the image is part of our current research. The issues of which relations to use, how to vary those relations with time, and how to incorporate the variations in object descriptions that occur over time are also being studied.

The image sequence is not the only source of data used by an interpretation system. A large database of relational and descriptive information, including information about processes and procedures, is also necessary for image interpretation [13,30]. These data are a symbolic representation of knowledge about the world, especially as that knowledge pertains to the interpretation problem; thus, the database is usually referred to as a knowledge base. For example, if it is known that the camera is upright, level, and outdoors, then it can reasonably be expected that the bright area at the top of the image is the sky. Such simple rules and other more complex inferences form the greater part of knowledge for interpretation. The problem of selecting and representing the appropriate knowledge is a difficult one. Understanding of image interpretation must be achieved by building systems of processes that work in restricted domains. In building such systems, the rules and processes which can eventually be incorporated into a knowledge base are learned. The problem then is to organize the information about the object and the procedures for using that information so that they can work in situations where the answers are less than certain.

The issues are basically these:

- Organization must be imposed on the information used for image interpretation, because that information is richly detailed and complex, and because large amounts of information are required.
- Varied types of information (for example, relations, procedures, and structural descriptions) need to be made easily accessible at many levels of abstraction.
- There must be a way to control the selection of which information is considered, which programs are activated, and what information is passed among programs.
- Interpretation processes need to deal with errors from image data and introduced by processes producing partial interpretation.
- All components of the interpretation system should be able to deal with the changing nature of dynamic data and the reorganization of interpretations that occur when a process creates new hypotheses.

6 Vision Computer Architecture

The real-time application of image understanding algorithms requires computer hardware that allows the algorithms to be executed at high speed. The NCUBE machine is being used to study hypercube architectures for the implementation of vision algorithms.

This research is concerned with developing techniques to perform computer vision computations on loosely and tightly coupled parallel processors. The size of data sets and the speed at which they can be created in current problems swamp even the fastest computers. The processing power required to keep up with the input is greater than 100 MIPS. This level of processing power is possible only with some degree of parallelism. However, a rule of thumb is that parallel processors can rarely sustain an efficiency level of greater than 20% [38]; thus the machine needs to have a peak power of about 500 MIPS. Normally the number of operations far exceeds this and the size of the input data set is often much greater than 512 times 512 bytes, either because multiple views are required, or because the sensor has a higher resolution (4096 times 4096, for example), or more than one modality is in use.

In the past, vision researchers have been forced to use SIMD meshes (such as the Goodyear MPP or pyramids) to achieve this rate of computation. Hypercubes offer a better alternative than SIMD meshes for several reasons.

- They can efficiently simulate these SIMD machines
- Their interconnections are better for operations such as image rotation, that are inefficient on meshes or pyramids
- The hypercube is better suited to higher level symbolic tasks since it is an MIMD machine, the nodes have far more memory, the interconnections are much better for the more random information exchange, and the node processors have a better instruction set

As an example of efficiency in vision tasks, a hypercube with p processors can rotate an n times n image 90° in $O(n^2 \log p/p)$ time, versus $O(n^2/\sqrt{p})$ for a mesh or pyramid.

Hypercube architectures for MIMD organizations are better than architectures with SIMD organizations for operations at the intermediate levels of vision processing where the sequence of operations for different regions in an image are usually different because the sequence is governed by properties of the region. For this type of processing, a loosely coupled multiprocessor system is ideal. If dynamic scenes are involved, then massive parallelism is essential [1].

7 Sensor Integration

Representations and techniques for combining vision, range, tactile, and other sensory information are being studied. Representations that will allow easy inference using multi-sensors are not known. Representations that will facilitate combining multi-sensor information for planning and recognition tasks using partial information obtained from each source are key to the success of sensor fusion. Techniques to combine uncertain and imprecise information are being developed using nonmonotonic and probabilistic approaches. Methods that try to combine beliefs and disbeliefs using uncertainty calculus for solving problems in distributed problem solving systems are being studied.

Computer vision and knowledge-based systems have to deal with uncertain and imprecise data in almost all phases of reasoning. Many approaches, such as Bayesian methods, fuzzy logic, Dempster-Schaefer theory, and qualitative approaches, have been proposed for dealing with uncertainty. All of these approaches have their problems and advantages. Unfortunately, it is not clear which method is best for which types of applications. With the aim to understand the strenghts and weaknesses of each approach, a language called ULOG is being developed that will implement uncertain variables in a PROLOG type environment. The ULOG language allows a user to change the algorithm used

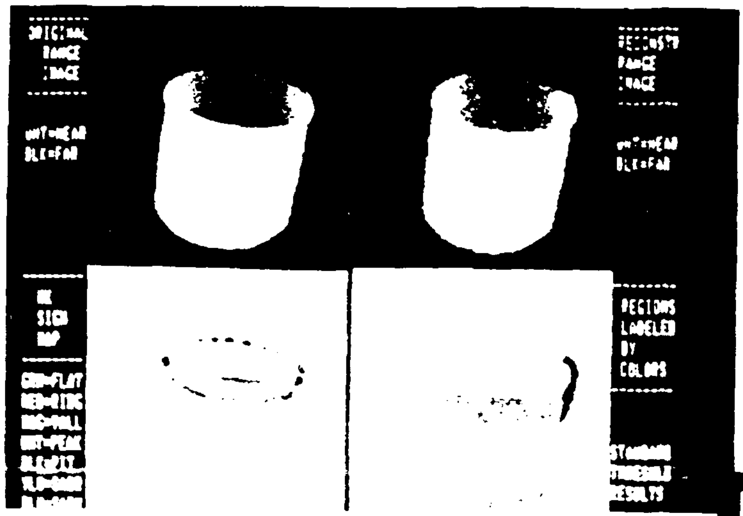


Figure 4: A range image and its segmentation using Besl's and Jain's algorithm are shown in the top-left and bottom-right corners respectively. Using the polynomial descriptors of the segmented surfaces, the original image is reconstructed. This is shown at top-right.

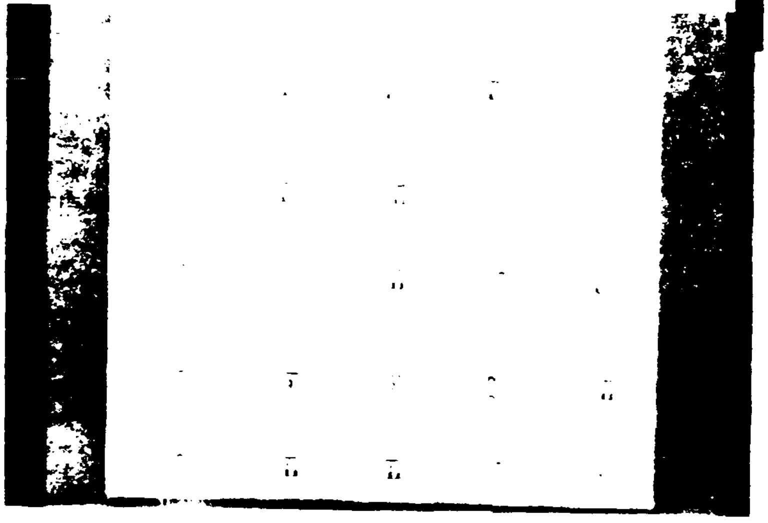


Figure 5: This figure shows an image showing the classification of solder joints by our algorithm.

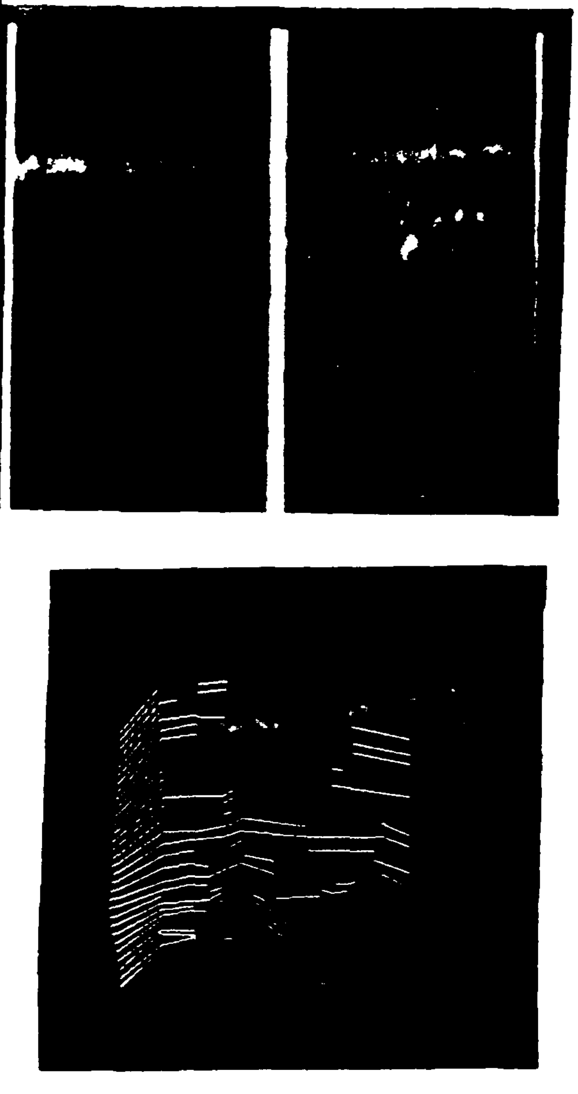


Figure 6: This figure shows an image showing two SEM images of a section of a wafer and the 3-D surface structure as reconstructed using our sem stereo algorithm.

for uncertainty management to experiment with different approaches in order to find the best approach for a given problem.

8 Applications

Several research projects are closely applied with specific applications. Normally, the research projects are focussed on the development of generic algorithms that can be applied to a wide variety of applications. All of the research work on generic algorithm can lead to applications. However, occasionally some application is investigated in the Computer Vision Research Laboratory because the application provides a focus for learning about some class of vision algorithms. Research projects that are closely associated with applications are discussed in the following sections.

8.1 Solder Joint Inspection

This project is for improving the reliability of solder joints and to reduce the cost of manufacturing by reducing the fault rate. Techniques are being developed for inspection of solder joints to identify not only good/bad joints but also to identify the nature of defects. By identifying the nature of defects and using trend analysis, the system may be able to indentify the process parameters that may result in defective joints [7]. An algorithm based on surface characteristics classifies the joints. The performance of the current version of the algorithm is 97-98% correct results. A knowledge-based approach to the task is being implemented to improve the performance of the classifier still further and to identify process parameters [3]. Some results from this project are shown in Figure 5. Recent extensions to this research project include inspection algorithms for circuit boards with surface mounted components.

8.2 Semiconductor Wafer Inspection

This project is a part of the center of excellence in semiconductor manufacturing. The project is concerned with the in-process inspection of semiconductor wafers to identify defects. The objective is to identify defects to control subsequent processes; if possible, the system will take corrective action and adjust the defective process parameters at the stage that is be responsible for the defect. The corrective actions are automatically performed by an expert system that controls the processes. The inspection is done with scanning electron microscopy (SEM) and optical microscopy. Current activities include the study of SEM images to implement stereo approaches for those images to perform analysis of those images to inspect the nature of edges and other features on a wafer. A model-based stereo algorithm has been implemented to reconstruct the 3-D structure of a wafer section. Figure 6 shows 3-d surface structure recovered by our algorithm. Knowledge-based controls are being developed for the interface between the vision system and the expert system in the automated semiconductor manufacturing project.

References

[1] Agrawal, D. P. and R. Jain, "A pipelined pseudoparallel architecture for dynamic scene analysis," *IEEE Trans. Computers* 31 (1982), 952-962.

[2] Barrow, H. G. and J. M. Tenenbaum, "Recovering intrinsic scene characteristics from images," *Computer Vision Systems.* ed. A. R. Hanson and E. M. Riseman. New York: Academic Press, 1978.

[3] Bartlett, S. L., C. L. Cole, and R. Jain, "Expert system for visual solder joint inspection," in *Proc. IEEE Conf. Art. Int. Applications*, March 1987.

[4] Besl, P. G. and R. Jain, "Segmentation through Symbolic Surface Descriptions," Accepted for publication in *IEEE Trans. Pattern Analysis and Machine Intelligence.*

[5] Besl, P.G. and R. Jain, "Invariant Surface Characteristics for 3-D object recognition in depth maps," *Comp. Vision, Graph. and Image Proc.* 33 (1986), 33-80.

[6] Besl, P.G. and R. Jain, "Three-dimensional object recognition," *Computing Surveys* 17 (1985), 75-145.

[7] Besl, P. G., E. Delp, and R. Jain, "Automatic visual solder joint inspection," *IEEE J. Robotics and Automation* 1 (1985), 42-56.

[8] Binford, T., "Survey of Model Based Image Analysis Systems," *International Journal of Robotics Research* 1 (1982), 18-64.

[9] Brady, M., "Computational Approaches to Image Understanding," *Computing Surveys* 14 (1982), 3-71.

[10] Brooks, R., "Symbolic Reasoning Among 3-D Models and 2-D Images," Technical Report STAN-CS-81-861, Department of Computer Science, Stanford University, Stanford, California, June 1981.

[11] Gibson, J. J., *The ecological approach to visual perception.* Boston: Houghton Mifflen, 1979.

[12] Grosky, W. I., and R. Jain, "A pyramid-based approach to segmentation applied to region matching," Accepted for publication in *IEEE Trans. Pattern Analysis and Machine Intelligence.*

[13] Hanson, A., et al. "A Methodology for the Development of General Knowledge-Based Vision Systems," Technical Report, COINS Department, University of Massachusetts, September 1985.

[14] Haynes, S. M. and R. Jain, "Detection of Moving Edges," *Comp. Vision, Graph. and Image Proc.* 21 (1983), 345-267.

[15] Haynes, S. M. and R. Jain, "Event detection and correspondence," *Optical Engineering* 25 (1986), 387-393.

[16] Horn, B. K. P. and B. G. Schunck, "Determining optical flow," *Artificial Intelligence* 17 (1981), 185-203.

[17] Horn, B. K. P., *Robot Vision*. Cambridge: M. I. T. Press, 1986.

[18] Jain, R., Bartlett, S., and N. O'Brien, "Motion Stereo Using Egomotion Complex Logarithmic Mapping," Accepted for publication in *IEEE Trans. Pattern Analysis and Machine Intelligence*.

[19] Jain, R., D. Militzer, and H.-H. Nagel, (1977) "Separating non-stationary from stationary scene components in a sequence of real world TV-images," in *Proc. Int. Joint Conf. Artificial Intelligence*, pp. 612–618.

[20] Jain, R. and H.-H. Nagel, (1979) "On the analysis of accumulative difference pictures from image sequences of real world scenes," *IEEE Trans. Patt. Anal. and Mach. Intel.* 1 (1979), 206–214.

[21] Jain, R., "Direct computation of the focus of expansion," *IEEE Trans. Patt. Anal. and Mach. Intel.* 5 (1983), 58–64.

[22] Jain, R. "Segmentation of frame sequences obtained by a moving observer," *IEEE Trans. Patt. Anal. and Mach. Intel.* 6 (1984), 624–629.

[23] Jain, R. "Difference and Accumulative Difference Pictures in Dynamic Scene Analysis," *Image and Vision Computing* 2 (1984), 99–108.

[24] Jain, R., T. Sripradisvarakul, and N. O'Brien, "Symbolic Surface Descriptors for 3-D Object Recognition," in *Proc. SPIE*, January 1987.

[25] Jain, R. and N. O'Brien, "Ego-Motion Complex Logarithmic Mapping," in *Proc. SPIE*, November 1984.

[26] Jerian, C. P. and R. Jain, "Determining motion parameters for scenes with translation and rotation," in *Proc. Workshop on Motion*, Toronto, 1983.

[27] Kayaalp, A. E. and R. Jain, "A knowledge-based automatic on line wafer inspection system," in *Proc. Vision 85*, pp. 5.117-5.130, 1985.

[28] Knoll, T. F. and R. Jain, "Recognizing partially visible objects using feature indexed hypothesis," *IEEE J. Robotics and Automation* 2 (1986), 3–13.

[29] Liou, S. P. and R. Jain, "Detecting Road Edges Using Hypothesized Vanishing Points," Accepted for publication in *Comp. Vision, Graph. and Image Processing*.

[30] Macworth, A., "Vision Research Strategy: Black Magic, Metaphors, Mechanisms, Miniworlds, and Maps," *Computer Vision Systems*. ed. A. Hanson and E. Riseman. New York: Academic Press, 1978.

[31] Neisser, U., *Cognition and Reality*. San Francisco: W. H. Freeman, 1976.

[32] O'Brien, N. G. and R. Jain, "Axial motion stereo," in *Proc. Workshop on Computer Vision*, Annapolis, Maryland, 1984.

[33] Rao, A. R., R. Jain, and T. E. Weymouth, "Knowledge-based vision systems," Submitted to *Computing Surveys*.

[34] Rao, A. R. and R. Jain, "Knowledge representation and control in computer vision systems," Accepted for publication in *IEEE Expert*.

[35] Schunck, B. G. "Motion segmentation and estimation," Doctoral Thesis, Department of Electrical Engineering and Computer Science, M. I. T., 1983.

[36] Schunck, B. G. "The motion constraint equation for optical flow," in *Proc. Int. J. Conf. Pattern Recognition*, pp. 20–22, Montreal, Canada, 1984.

[37] Schunck, B. G. "Motion segmentation and estimation by constraint line clustering," in *Proc. Workshop on Computer Vision*, pp. 58–62, Annapolis, Maryland, 1984.

[38] Schunck, B. G. "Surface-based smoothing of optical flow fields," in *Proc. Conference on Intelligent Systems and Machines*, pp. 107–111, Oakland University, Rochester, Michigan, 1984.

[39] Schunck, B. G. and B. K. P. Horn, "Constraints on optical flow computation," in *Proc. Conf. Pattern Recognition and Image Processing*, pp. 205–210, 1981.

[40] Schunck, B. G. "Image Flow: Fundamentals and Future Research," in *Proc. Conf. Computer Vision and Pattern Recognition*, San Francisco, 1985.

[41] Sethi, I.K. and R. Jain, "Finding trajectories of feature points in a monocular image sequence," *IEEE Trans. Patt. Anal. and Mach. Intel.* 9 (1987), 56–73.

[42] Sethi, I.K., "A fast algorithm for recognizing nearest neighbors," *IEEE Trans. Systems, Man, and Cybernetics* 11 (1981), 245–248.

[43] Sethi, I.K. and R. Jain, "Determining three dimensional structure of rotating objects using a sequence of monocular views," Technical Report, Wayne State University, 1984.

[44] Shah, M. A. and R. Jain, "Detecting Time Varying Corners," *Computer Vision, Graphics, and Image Processing* 28 (1984), 345–355.

[45] Ullman, S. *The Interpretation of Visual Motion*. Cambridge: M. I. T. Press, 1979.

[46] Weymouth, T. E., J. S. Griffith, A. R. Hanson, E. M. Riseman, "Rule Based Strategies for Image Interpretation," in *Proc. AAAI*, pp. 429–432, August 1983.

[47] Weymouth, T. E., "Using Object Descriptions in a Schema Network for Machine Vision," Ph. D. Dissertation, University of Massachusetts, April 1986.

AFOSR-TR- 89 1600

AIAA-87-1676 NASA Systems Autonomy Demonstration Project: Development of Space Station Automation Technology

J. S. Bull, R. Brown, P. Friedland,
C. M. Wong, W. Bates, K. J. Healy and
P. Marshall, NASA Ames Research Center,
Moffett Field, CA

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

March 9-11, 1987/Arlington, VA

NASA SYSTEMS AUTONOMY DEMONSTRATION PROJECT: DEVELOPMENT OF SPACE STATION AUTOMATION TECHNOLOGY

John S. Bull,* Richard Brown,** Peter Friedland,† Carla M. Wong,‡ William Bates,‡ Kathleen J. Healey,‡ and Paul Marshall‡

Abstract

Congress has displayed substantial interest in accelerating the dissemination of advanced automation technology throughout United States industries. The Space Station was selected as the high-technology program to serve as a highly visible demonstration of advanced automation, and spur dissemination of the technology to the private sector. NASA has recently initiated an Automation and Robotics Program to serve as the principal Research and Technology program contributing to Space Station automation.

The potential benefits of autonomy in terms of increased capability, safety, reliability, efficiency, and cost savings for operational systems on the Space Station has given strong impetus to the development of a Project for feasibility demonstrations of automation technology. Additional strong motivation was given by the report and recommendations to Congress by the Advanced Technology Advisory Committee (ATAC): "The development of the Space Station offers a chance, both to advance the technology of automati and robotics as proposed by Congress and to put the technology to use. The use of advanced automation and robotics technology in the Space Station would thereby provide a logical driving force for a new generation of machine intelligence, robotics, computer science, and microelectronics."

The NASA Systems Autonomy Demonstration Project has been initiated in response to this Congressional interest and ATAC recommendations for Space Station automation technology demonstrations. The demonstrations will begin with the testbed for the Space Station Thermal Control System (TCS) in 1988. Additional demonstrations are scheduled for 1990, 1993, and 1996. The 1990 demonstration will involve coordinated control of two Space Station subsystems through cooperating expert systems, the 1993 demonstration will involve hierarchical automation of multiple subsystems, and the 1996 demonstration will involve distributed automation of multiple subsystems.

The 1988 TCS Demonstration will be a joint cooperative effort between NASA Ames Research Center (ARC) and NASA Johnson Space Center (JSC). Knowledge-engineering and operator-interface technologies for Systems Automation will be developed by knowledge engineers, AI researchers, and human factors researchers at ARC by relying on a close working relationship with the domain experts, knowledge and integration engineers, and mission operations personnel at JSC.

This paper provides an overview of the NASA Systems Autonomy Demonstration Project, and describes general details of the 1988 TCS Demonstration including expert system functional objectives, system concept, and development plans.

Introduction

In keeping with the mandates of the National Aeronautics and Space Act of 1958 and the National Space Strategy approved by the President and Congress in 1984, NASA has set for itself a major goal of "conducting effective and productive space applications and technology programs which contribute materially toward U.S. leadership and security."

The Report of the National Commission on Space (May 1986) in its vision of the next 50 years on space strongly recommends an integration of humans and machines through automation and robotics. Specifically it is recommended that "NASA explore the limits of expert systems, and tele-presence or tele-science for remote operations, including ties to spacecraft and ground laboratories."

Congress has displayed substantial interest in accelerating the dissemination of advanced automation technology within United States industries. Space Station was selected as the high-technology program to serve as a highly visible demonstration of advanced automation, and spur dissemination of the technology to the private sector. NASA has recently initiated an Automation and Robotics Program to serve as the principal Research and Technology program contributing to Space Station automation.

The potential benefits of autonomy in terms of increased capability, safety, reliability, efficiency, and cost savings for operational systems on the Space Station has given strong impetus to the development of a Project for feasibility demonstrations of automation technology. Additional strong motivation was given by the report and recommendations to Congress by the Advanced Technology Advisory Committee (ATAC): "The development of the Space Station offers a chance, both to advance the technology of automation and robotics as proposed by Congress and to put the

*Aerospace Engineer. Member AIAA.
**Aerospace Engineer.
†Research Scientist.
‡Engineer.

technology to use. The use of advanced automation and robotics technology in the Space Station would thereby provide a logical driving force for a new generation of machine intelligence, robotics, computer science, and microelectronics."

The NASA Systems Autonomy Demonstration Project has been initiated in response to this Congressional interest and ATAC recommendations for Space Station automation technology demonstrations. The demonstrations will begin with the testbed for the Space Station thermal control system (TCS) in 1988. Additional demonstrations are scheduled for 1990, 1993, and 1996. The 1990 demonstration will involve coordinated control of two Space Station subsystems through cooperating expert systems, the 1993 demonstration will involve hierarchical automation of multiple subsystems, and the 1996 demonstration will involve distributed automation of multiple subsystems.

The 1988 TCS Demonstration will be a joint cooperative effort between NASA Ames Research Center (ARC) and NASA Johnson Space Center (JSC). Knowledge-engineering and operator-interface technologies for Systems Automation will be developed by knowledge engineers, AI researchers, and human factors researchers at ARC by relying on a close working relationship with the domain experts, knowledge and integration engineers, and mission operations personnel at JSC.

The TCS Automation involves the implementation of current AI technology into the real-time dynamic environment of a complex electrical-mechanical Space Station system. The TCS includes real-time nominal control, fault diagnosis and correction of real-time problems, design and reconfiguration advice on the thermal testbed (TTB), and an intelligent interface to both novice and expert users. The TCS Demonstration will accelerate the transfer of Systems Autonomy research technologies to user applications in a real-time operational environment, and increase user confidence in the new technologies.

Systems Autonomy Demonstration (SADP) Project

Objectives

The objectives of the Systems Autonomy Demonstration Project are to provide:

1. Technical base of in-house personnel and development tools to facilitate AI technology transfer
2. Technology focus for Automation Research and Development in support of NASA's Space Programs
3. Means for validation and demonstration of Automation Technology prior to transfer to Agency programs
4. Credibility of Automation Technology within NASA
5. Credibility of NASA AI expertise to the outside AI community

Broad Approach

The Systems Autonomy Demonstration (SADP) will be a joint, cooperative effort between research and operational NASA Centers. The required AI technologies will be developed and implemented by knowledge engineers, and AI and human factors researchers at ARC; while relying upon the domain experts, knowledge and integration engineers, and mission operations personnel residing at operational NASA Centers.

The SADP project approach will involve a multidisciplinary integration of knowledge engineering, man/machine interfaces, and systems architecture to enhance automation of the Space Station.

Each project demonstration will proceed through a phased knowledge engineering methodology consisting of: prototype knowledge base development, incremental knowledge base expansion, parallel operator interface development, and implementation in a realistic environment. Initial AI system development will be carried out in-house so as to develop a strong technical base within NASA.

Demonstrations will involve interaction with both experts and novice personnel representing mission operations, automated flight subsystems, and automated sciences. This interaction with astronauts and ground crew is a critical component of the project to ensure that the human interface (man-machine) issues are properly addressed.

SADP Demonstration Selection Criteria

The following criteria will be used in selecting demonstrations:

1. Provide maximum use of existing AI technologies
2. Illustrate gains in human productivity and reductions in manpower requirements resulting from automation
3. Have access to domain experts
4. Requires no unattainable personnel and equipment resources
5. Leave a framework of people and tools which will facilitate future technology transfer
6. Transfer developed technology to the Space Station

SADP Management

The information Sciences and Human Factors Division of NASA's Office of Aeronautics and Space Technology provides overall direction, funding, and evaluation of the Systems Autonomy Demonstration Project being managed by the Systems Autonomy Demonstration Project Office in the Information Sciences Office at ARC.

Thermal Control System (TCS) Demonstration

TCS Selection Rationale

The rationale for selecting TCS as the Demonstration Project is that it meets the above SADP criteria and provides:

1. Guaranteed access is available to Domain Experts
2. TCS slow dynamics reduce technical risks
3. Adequate personnel and resources are available
4. Demonstration schedule matches well with TTB
5. Environment for interface with Space Station Data Management Testbed

TCS Demonstration Technical Objectives

Technical objectives of TCS Automation are to demonstrate:

1. Real-time nominal control and reconfiguration for four to five failure modes
2. Fault diagnosis of 25 to 30 major failure modes
3. Trend analysis incipient failure prevention
4. Intelligent interface to both novice and expert users
5. Design advice on TTB
6. Training assistance

Technology Thrusts

The major technology thrusts of the TCS Demonstration are:

1. Integration of knowledge base systems into a real-time environment
2. Causal modeling of complex components and elements
3. Combining model-based and experiential knowledge for diagnosis
4. Trend analysis heuristic rules
5. AI validation methodologies

TCS Automation Benefits

The major benefits of TCS Automation are:

1. Reduces need for crew and ground monitoring of TCS
2. Increases crew safety through improved systems monitoring
3. Provides TCS design assistance
4. Simplifies novice- and expert-user training

TCS Management Organization

The Thermal Control System (TCS) Demonstration Project is managed by the SADP Office at ARC in a close working relationship with Aerospace Human Factors Division at ARC, the Crew and Thermal Systems Division at JSC, the Systems Development and Simulation Division at JSC, and the Mission Operations Directorate at JSC.

TCS General Approach

The Space Station TTB is being developed by the Crew and Thermal Systems Division at JSC from which the domain expertise is being provided. The knowledge engineering and demonstration prototype development are being done by the SADP Office and the Artificial Intelligence Research Branch with support from the Aerospace Human Factors Division at ARC. The Systems Development and Simulation Division at JSC provides support and participates with the SADP Office at ARC in the knowledge engineering and expert system development aspects of the TCS project and provides testbed integration. The Mission Operations Directorate of JSC provides consultation and advice on recent trends and technology advancements in operations' automation and the application of these technologies and current mission operations' philosophy to the TCS. The general TCS approach is shown in Fig. 1.

Facilities

The major facility required for the TCS Demonstration is the TTB being constructed at JSC. The TTB includes the following subsystems: 1) Thermal System Test Articles (pumps, radiators, evaporators, condensers, busses), and 2) a Data Acquisition and Control Ssytem.

A facility is located at ARC for development of the Thermal Expert System (TEXSYS). This facility will consist of 1) AI HW/SW development tools and 2) a simulation HW/ model of the TTB for TEXSYS development and validation.

TCS Schedule

The broad overall TCS Demonstration Project Schedule is shown in Fig. 2

Success Criteria

Programmatic success criteria, although more difficult to define than technical success criteria (meeting system specifications), is of equal importance. These criteria are the incorporation of systems autonomy technology (developed as a result of and demonstrated during the TCS Demonstration) in various Space Station subsystems and systems. This criteria does not imply direct incorporation of TEXSYS, or any part thereof. Rather, it implies an influence on Space Station Project Offices, measured by the incorporation of autonomy requirements in subsystem requirements documents and the inclusion of automation in the design and development of those subsystems.

Discussion

This section discusses the TCS System Concept, Specific Technical Objectives, Development Plan, and Status.

General Thermal System Requirements

The Thermal Control System provides thermal management of most Space Station elements through heat acquisition, transportation, and rejection. A schematic of the baseline two-phase Space Station TCS is shown in Fig. 3. General system requirements are:

1. Narrow-band temperature control among all service areas
2. Long-distance transport of waste heat
3. Multi-year service reliability
4. Reconfigurable heat source operation
5. On-orbit growth capability to satisfy Space Station requirements

Specific Thermal Control Expert System Objectives

The Thermal Control Expert System (TEXSYS) knowledge-based functions within the ECS are:

1. Diagnosis and Fault-Correction Advice
2. Incipient Failure Prevention Through Trend Analysis
3. Nominal Real-Time Control and Fault Correction
4. Intelligent Interface to Novice and Expert Users
5. Training Assistance
6. Design Assistance

Diagnosis and Fault Correction Advice. The TCS can have faults in three major categories: components, control malfunctions, and sensor malfunctions. The TCS demonstration will show expert-level ability to diagnose and suggest corrective actions on approximately 25 to 30 common TCS faults, representing essentially all major modes of TCS failures.

Incipient Failure Prevention. Human beings are notoriously poor at the slow and careful analysis that is needed to prevent low frequency dynamic anomalies from escalating into problems. A potential strength of a knowledge-based systems approach to thermal management is the use of trend analysis to detect long-term degradation and reconfiguration as required to prevent system parameters from exceeding operational limits. The demonstration will exhibit "offline" (i.e., during noncrisis times) analysis of trends to detect anomalous values to make corrections to the thermal system before serious problems result.

Real-time Control and Fault Correction. The demonstration will exhibit real-time nominal control as well as real-time correction of at least four or five major failure classes of the thermal system. In the context of the thermal system, real-time is a matter of seconds. The TCS expert system will analyze actual sensor data, notice and diagnose problems, and correct (or bypass) problems by sending control signals to the thermal system.

Intelligent Interface. The demonstration will show the ability of the knowledge-based TCS expert system to explain its reasoning to users. The operator interface will allow users access to information on testbed schematics, all stages of fault reasoning, basic physical principles underlying component and TCS system behavior, and provide guidance in making decisions involving thermal management. The interface will be a "direct manipulation" style interface, combining mouse-based pointing and menu selection as user input; and the interface will show some degree of understanding of the skill level of its user.

Training Assistance. A beneficial side effect of knowledge-based systems is that the knowledge bases have substantial utility for future training purposes with the system. The information display capabilities will demonstrate how the knowledge-based TCS expert system can be used for purposes of crew and ground operator training in the context of Space Station. Trainees will be able to examine data and simulate the effects of all known faults.

Design Assistance. The expert system modeling and simulation knowledge base provides a substantial capacity for intelligent assistance to the design engineer using the TTB. The information and display capabilities will demonstrate the ability to automatically reflect new physical realities resulting from design changes during system configuration change investigations.

TEXSYS Conceptual Configuration with the TTB

The conceptual configuration of TEXSYS within the TTB is shown in Fig. 4.

Individual test articles are directly connected to control computers (microVAXes) which then connect to an Ethernet. A DACS computer (a larger microVAX II) acts as a system controller, central data router, and command queuer for the TTB. The TEXSYS initially running on a specialized LISP machine will be connected via standard DECnet protocols to the Ethernet. The machine may receive data from the DACS system and pass commands to the DACS system. If this routing strategy is not sufficiently fast or powerful, data will be received from and commands passed directly to the test article controllers.

Another possibility for increasing speed would be the use of a conventional computer as a front-end processor (FEP) along with a LISP machine. Such an arrangement could be useful if the major speed bottleneck turns out to be in handling the raw data from DACS. The FEP could handle preprocessing tasks to reduce the raw data to a compressed amount of information that can be handled by the expert system. It might be possible that the DACS itself could provide this data reduction. Further investigation of this area will be necessary.

TEXSYS System Development Environment

The initial version of TEXSYS will make use of the KEE knowledge-based building tool (from IntelliCorp, Inc.) and the ZetaLisp development environment (from Symbolics, Inc.) running on one of the Symbolics family of LISP workstation computers. User-interface design will make use of some combination of the standard Symbolics bitmap display and an attached color graphics workstation. The Symbolics ZetaLisp environment provides simple mechanisms to directly call the FORTRAN subroutines that are used to send the proper information requests and reconfiguration commands out to subsystem controllers via the Ethernet.

TEXSYS Knowledge Base

The TEXSYS knowledge base will rely to a large degree on both experiential heuristic rules and causal or model-based reasoning. The initial TEXSYS concept will use a frame-based, hierarchical, object-oriented representation of knowledge. Frame-based means that each structural component of a thermal system, or each class of components, is represented by a collection of quantitative and qualitative facts about the component. Hierarchical means that each entity is not represented as a separate item, but as a tree of structures. Object-oriented means that both factual and procedural knowledge are accessed through the same mechanisms. Figure 5 shows examples of how the knowledge base is subdivided into thermal rules, systems, and component models.

TEXSYS Modeling and Simulation

The basis of all modeling and simulation in TEXSYS is the structural and functional knowledge base. Part of the knowledge-based construction task is to include pointers to relevant, existing mathematical models, to first principles of thermal engineering, and to heuristics for parameter propagation, for all components and subsystems involved in thermal management. Simulation in any of those cases proceeds in a straightforward, object-oriented manner. This means that a "simulate yourself" message gets passed to the relevant structure that is to be modeled and procedural knowledge of the appropriate form is activated. For qualitative models, this process is normally a call to a FORTRAN subroutine; however, for qualitative causal models, software tools convert laws of thermal sciences into actions; and for heuristic propagation of parameters, forward chaining is normally satisfactory.

The most difficult technical task will be the third step (the first two being description of the structural and functional knowledge, and developing inference methods to use that knowledge). This third task is selection of which of the three types of models (heuristic, qualitative, or quantitative) to use for any simulation, and to select the appropriate combination of information from different models. The selection itself will almost surely be heuristic, based upon expert knowledge of the relevance and trustworthiness of the various types of models in different situations. Combinations of models, especially when we are attempting to combine quantitative and qualitative knowledge, will be a significant research task. Most of the work will be experimental, testing various models in many different situations and determining the relevant speed accuracy/cost tradeoffs that apply.

General Development Plan

In any knowledge engineering project the work proceeds by incremental refinement of a relatively simple system, adding knowledge and consequently ability for increased performance; and by carrying out research in how to better combine types of knowledge and reasoning methodologies.

As one of the largest knowledge engineering projects yet attempted, this demonstration will use the previously described approach, and will also proceed along traditional project development methods: definition of the problem, specification of system requirements, definition of system specifications, development, validation, integration, checkout, and demonstration.

Specific Development Plan

The development and demonstration of the TEXSYS system will be accomplished through six major stages, most of which are separated by major project reviews. These stages include:

1. Prototype Development Stage
2. Requirements Definition Stage
3. System Specification Stage
4. Initial-System Development Stage
5. Final-System Development Stage
6. Demonstration Stage

Prototype Development Stage. As a first step in the problem definition and incremental engineering process, a small but significant prototype was constructed in June and July of 1986. The objectives of the prototype development were:

1. To learn significantly more, directly from an expert, about the TTB environment and about thermal engineering, especially as related to two-phase thermal systems on Space Station
2. To provide knowledge engineering training for ARC RI SADP personnel in a practical, problem-oriented environment
3. To build a working prototype system that would serve as a starting point for future work

An analysis was made of probable TEXSYS functional and performance requirements, available hardware, software, expert-system building tools, and training and engineering support. Based on this analysis, a selection was made as to the hardware and software to be used for the prototype development, and for the TEXSYS demonstration.

The work was done as a cooperative apprenticeship program under contract to a knowledge-engineering company, which provided several highly experienced knowledge engineers to assist in prototype development. All of the prototype demonstration objectives were accomplished successfully, and the prototype system was demonstrated to the SADP InterCenter Working Group (peer review group) in July 1986.

System Requirements Definition Stage. The next step in the development and demonstration of the TEXSYS system is the formal and specific definition of requirements. Particular care will be paid to interfaces to the operator, to the TCS Testbed computers, to real-time data collection and TEXSYS performance requirements, and to the architectural structure of the TEXSYS knowledge base. These features will be documented in the TEXSYS System Requirements Definition and will be reviewed at the TEXSYS System Requirements Review.

System Specification Stage. Following the successful accomplishment of the SRR, work will shift to the generation of a system design, including design of the knowledge base architecture, specific interfaces with necessary utilities, other systems, and the human operator or user of the TEXSYS, and the structure and format of data to be used as real-time input and output. This stage will be documented in the TEXSYS System Design Specification which will also specify the delivery hardware and software. This stage is completed at PDR.

Initial System Development Stage. The initial TEXSYS development activities will consist of procurement of test and demonstration hardware and software, and include two major phases of knowledge base development. The development activities at ARC will concentrate on development of TEXSYS knowledge bases and the human interface to TEXSYS, while JSC will take the lead in developing the software needed to integrate and interact with the real-time systems with which the TEXSYS will interface.

Phase one of the knowledge base development will consist of the acquisition and organization of knowledge about the TCS Testbed components and topology, and the development of rules for detecting and diagnosing problems. During this phase a static knowledge base will be used for testing purposes.

Phase two will include the modification of the system to successfully accommodate real-time operation and the provision of simulated dynamic data to test this major enhancement. This stage is complete at CDR.

Final System Development. After completion of the CDR and delivery, installation, and checkout of the TCS Demonstration software at JSC, the final development stage will begin. During this stage, the TEXSYS system will be completed and validated. The focus of development, integration and validation activities will be at the JSC facility; but with strong reliance on the technical cognizance of the Systems Autonomy Demonstration Project Office knowledge engineers for the demonstration system.

The system validation and integration activities at JSC shall be divided into three activities. First, the system that is delivered by ARC shall be tested using the Verification and Validation Plan.

Next, the knowledge base of the expert system shall be expanded in conjunction with the ARC knowledge engineers, the domain experts, and the integration experts to improve the knowledge representation, the domain expertise, and the operational competence of the expert system. In addition, when the final test and demonstration configuration of the TTB is selected, the expert system will have the knowledge base adapted to meet this configuration by joint efforts from the knowledge engineers and human factors personnel from ARC with the domain experts and integration experts at JSC.

Finally, the verification and validation tests shall be performed on the final demonstration TTB configuration jointly by ARC and JSC to ensure that the expert-system knowledge base is complete and correct. After passing these tests, TEXSYS shall be considered ready for the SADP 1988 demonstration phase.

This phase of the TEXSYS development will conclude with the TCS Operational Readiness Review (ORR). The ORR will examine all TMS Demonstration activities to determine the readiness of the system, the procedures and documentation, and the personnel for the conduct of the operational phase of the TCS Demonstration.

TCS Demonstration Stage. After successful completion of the ORR, the final phase, the Demonstration Phase, will begin. This stage, conducted jointly by ARC and JSC, will include the demonstration operations, and the post demonstration analysis and review.

The operations stage will involve the actual conduct and documentation of the TEXSYS in management and control of the TCS Testbed. The analysis and review phase will provide an integrated retrospective analysis of the system capabilities, and the development process, to provide insight into the effectiveness of the TEXSYS in management and control of the TCS Testbed and to identify improvements that can be made in later phases of the SADP project activities.

Formal Reviews

Formal project reviews will be conducted at appropriate points in the design and implementation of the TCS Demonstration. Five major reviews have been identified for the TCS Demonstration as follows, with their estimated schedule dates:

1. System Requirements Review (SRR-2/87)
2. Preliminary Design Review (PDR-5/87)
3. Critical Design Review (CDR-9/98)
4. Operational Readiness Review (ORR-7/88)
5. TCS Demonstration Review (TDR-10/88)

Concluding Remarks

Congress has displayed substantial interest in accelerating the dissemination of advanced automation technology to and in United States industry.

The NASA Systems Autonomy Demonstration Project has been initiated in response to this desire by Congress and will conduct Space Station automation technology demonstrations in 1988, 1990, 1993, and 1996.

The initial demonstration in 1988 involves automation of the Space Station Thermal Control System. An initial expert system prototype has been developed at ARC and the development is progressing on schedule for a demonstration on the Thermal Testbed at JSC in August 1988.

Acknowledgment

We would like to acknowledge the contributions to this project by the following persons: William Erickson, Renate Roske-Hofstrand, Roger Remington, and Mary Schwartz.

References

[1]Editor, "Advanced Automation and Robotics Technology for the Space Station and for the U.S. Economy." NASA TM-87566, 1985.

[2]National Commission on Space, "Presidential Space Study," July 1986.

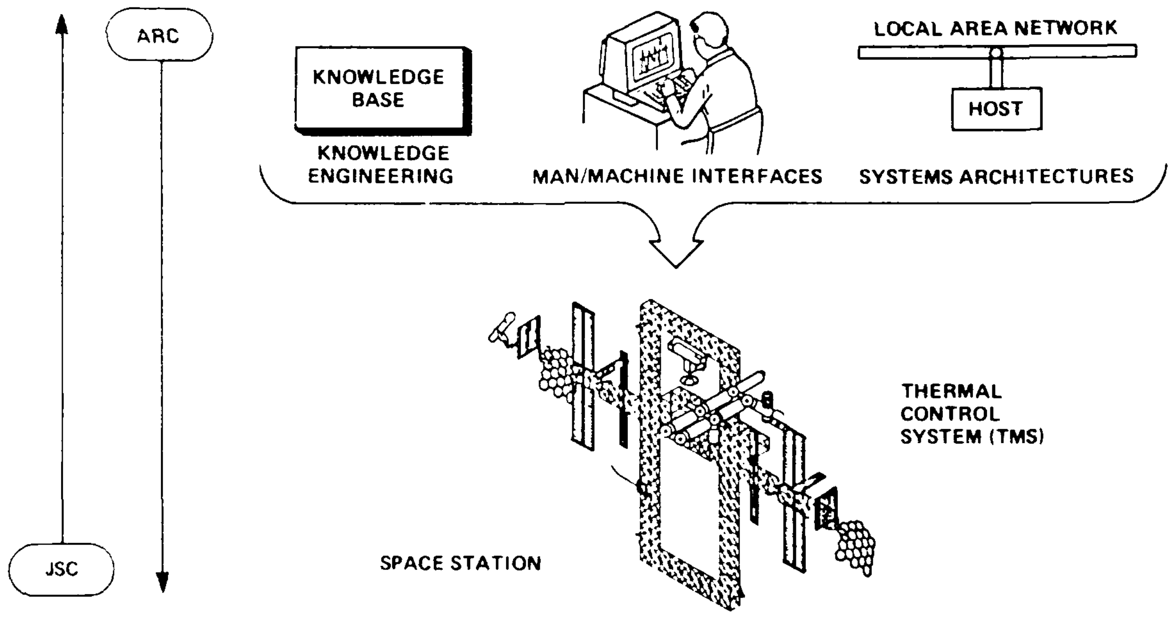
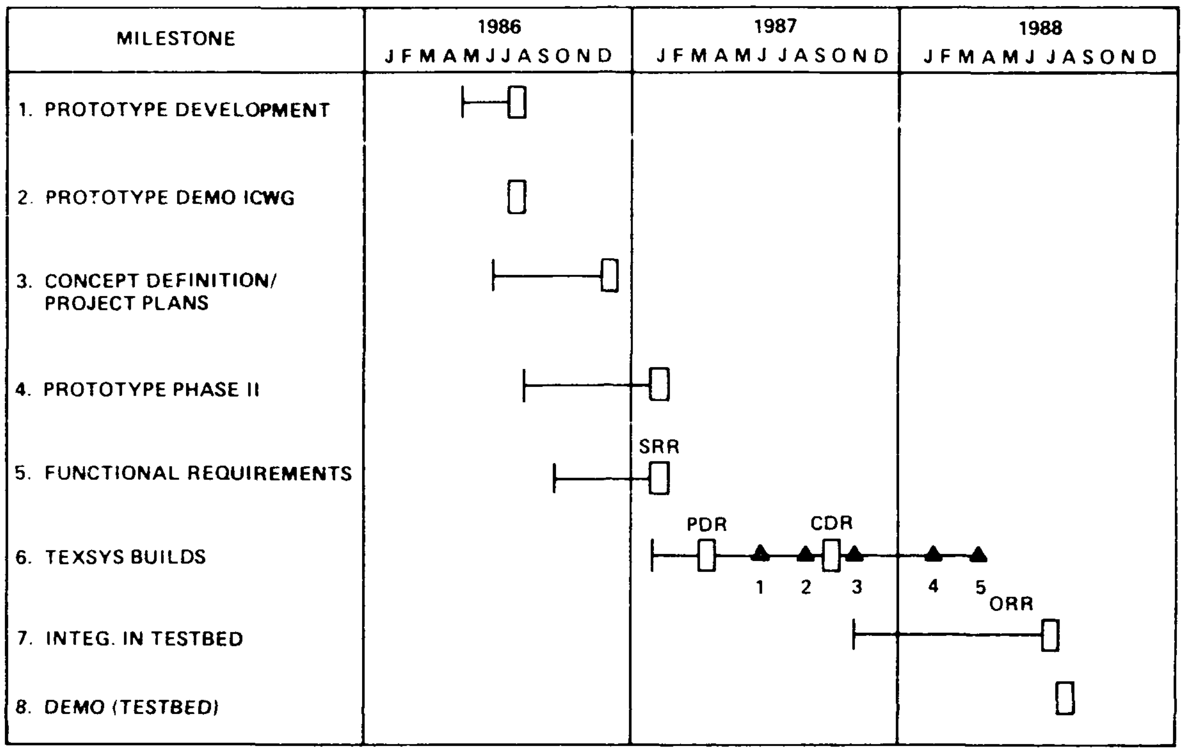


Fig. 1 TCS general approach.



NOTES: SSR - SYSTEM REQS. REVIEW
ORR - OPERATIONAL READINESS REVIEW
PDR - PRELIMINARY DESIGN REVIEW
CDR - CRITICAL DESIGN REVIEW
▲ - SYSTEM BUILD CAPABILITIES

Fig. 2 TCS broad schedule.

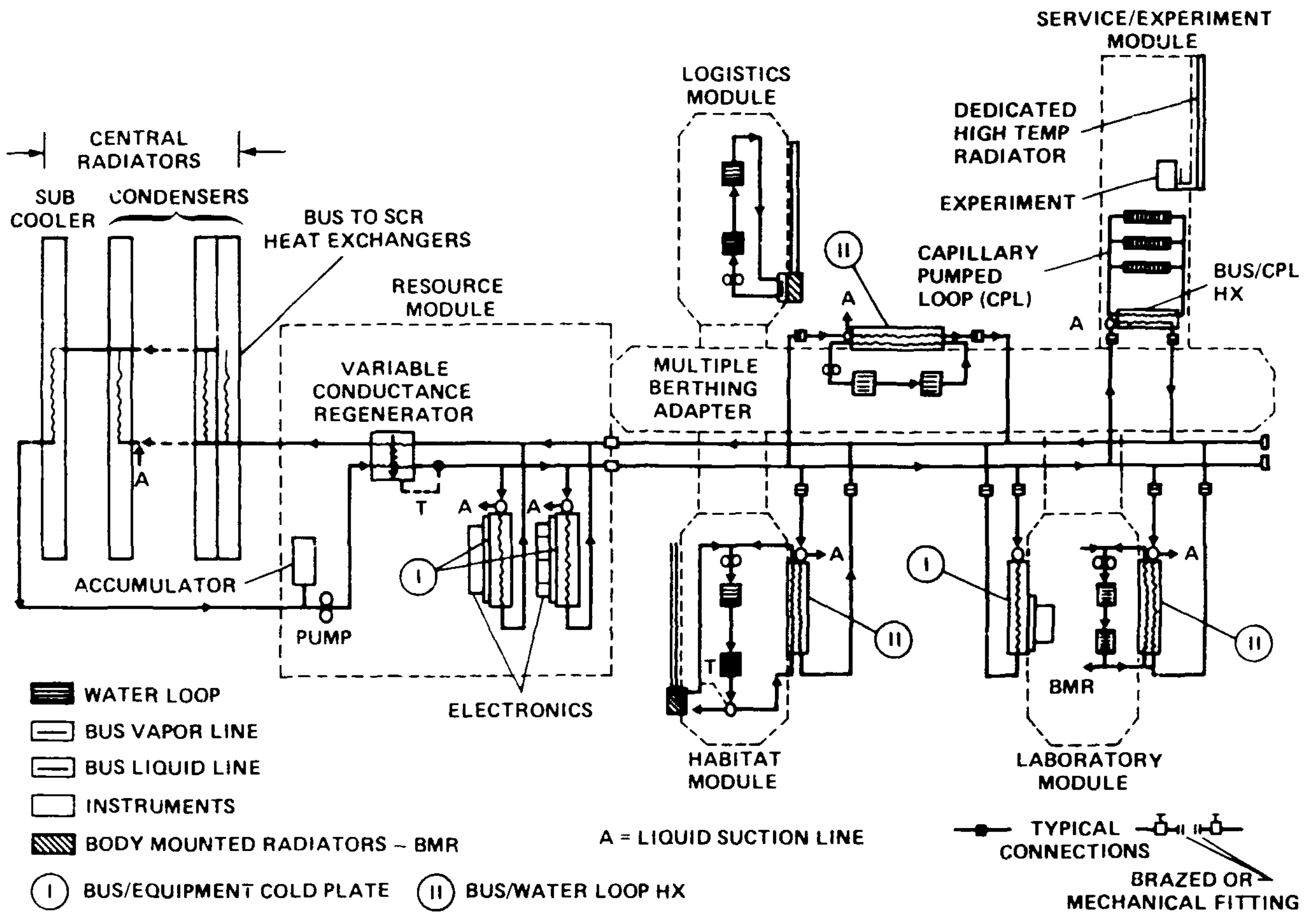


Fig. 3 Schematic of Space Station thermal management.

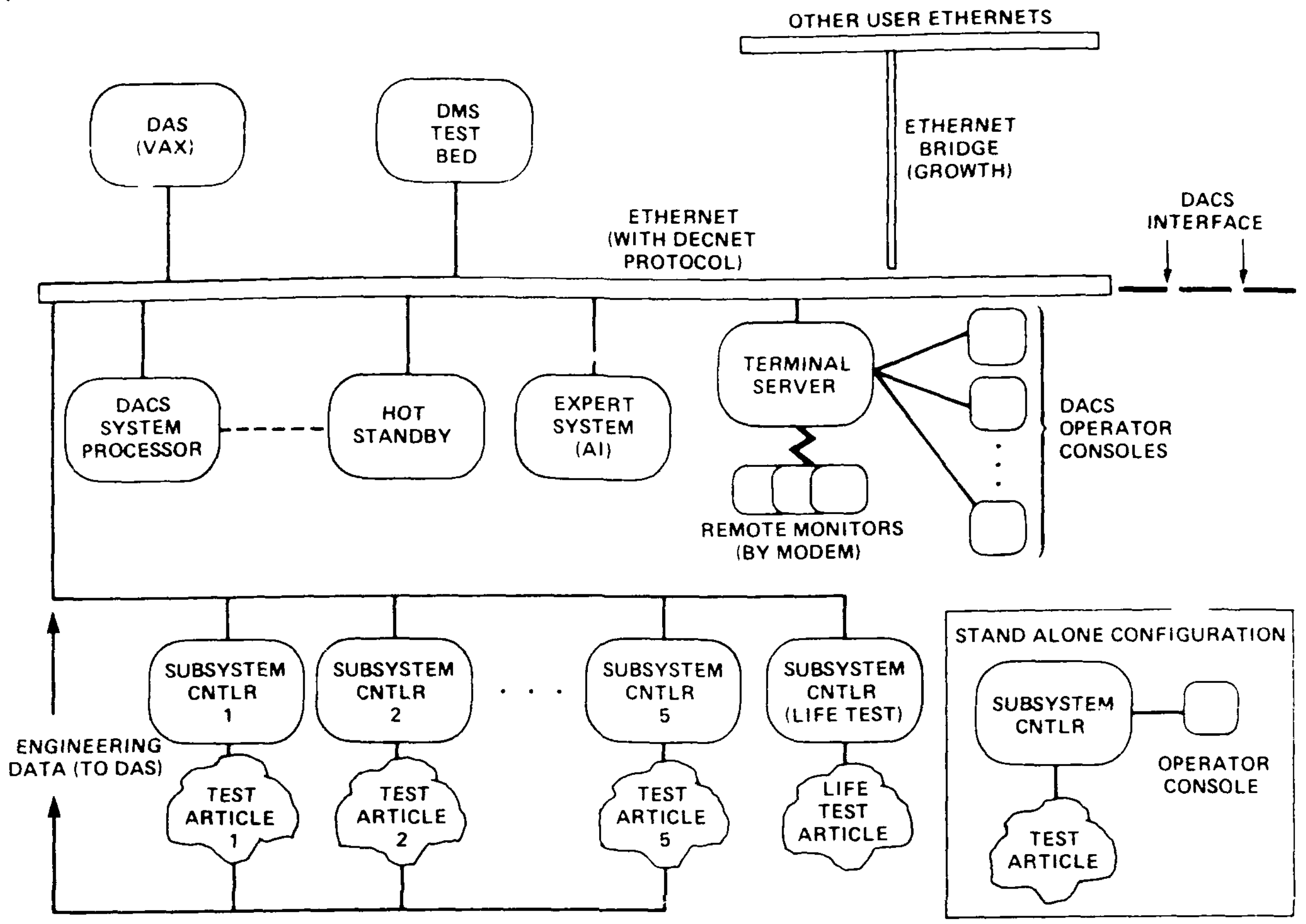


Fig. 4 TEXSYS conceptual configuration in TCS Testbed.

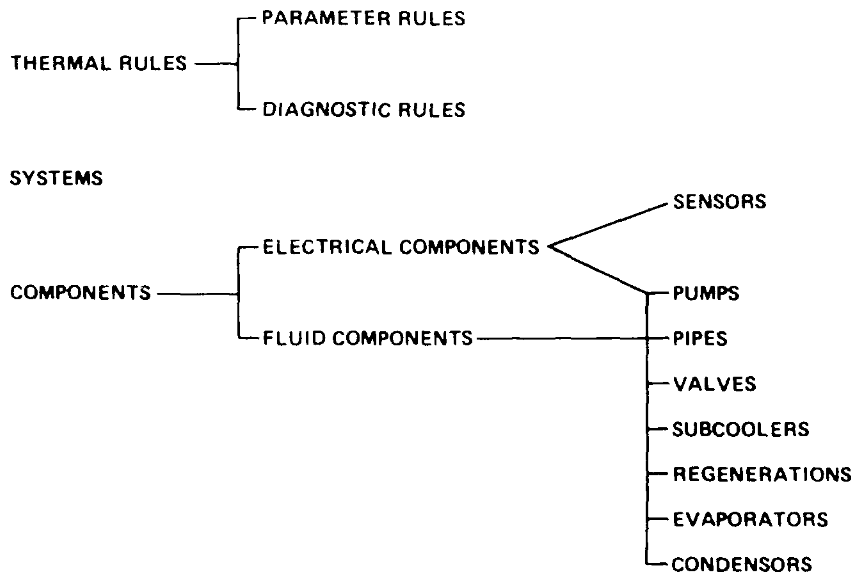


Fig. 5 TEXSYS knowledge base example.

AFOSR-TR- 88-1606

AIAA-87-1678
Hierarchical Classification: Its Usefulness for Diagnosis and Sensor Validation

B. Chandrasekaran and W. F. Punch III,
The Ohio State Univ.,
Columbus, OH

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

March 9-11, 1987/Arlington, VA

Hierarchical Classification: Its Usefulness for Diagnosis and Sensor Validation[1]

B Chandrasekaran and W F Punch III

Laboratory for Artificial Intelligence Research
The Ohio State University

Abstract

A number of views for the development of problem solving systems have emerged over the last decade in AI This paper will examine one such view, the concept of the *generic task*, one of a small set of ubiquitous cognitive tasks that together account for some part of human cognition In particular, we will show how *hierarchical classification* is one such generic task and how it is useful for diagnosis. We will also show that a diagnostic system based on hierarchical classification naturally lends itself to solving a number of issues traditionally associated with *sensor validation*

1. Introduction

The current generation of expert system languages -- those that are based on rules, frames, or logic -- do not distinguish between different types of knowledge-based reasoning For example, one would expect that the task of designing a car would require significantly different reasoning strategies than the task of diagnosing a malfunction in a car However, these methodologies apply the same strategy (fire the rules whose conditions match, run resolution engine on all propositions etc) to both design and diagnosis, as well as any other task Because of this, it has been argued that these methodologies, although useful, are rather low level with respect to modeling the needed task-level behavior In essence, these systems resemble an assembly language for writing expert systems While obviously useful, clearly approaches that more directly address the higher level issues of knowledge-based reasoning are needed for the next generation of AI development

One example of a higher level approach is the *generic task* [6]. The aim here is to identify "building blocks" of reasoning types such that each of the types is both generic and widely useful as components of complex reasoning tasks We have identified to date six such generic strategies, which together account for a very large portion of current expert system capabilities Each generic task is characterized by

1 The kinds of information required as input for the task and the information produced as a result of performing the task

2 A way to represent and organize the knowledge that is needed to perform the generic task.

3 The process (algorithm, control, problem solving) that the task uses

As each task and its associated structure is identified, languages are developed that encode both the problem solving strategy and knowledge that is appropriate for solving problems of that type These languages facilitate expert system development by giving the knowledge engineer access to tools which work at the level of the problem, not the level of the machine Below is a list of the generic tasks that have already been identified and the tools that correspond to them:

1 *Hierarchical Classification* is finding the categories in a classification hierarchy that apply to the situation being analyzed The tool for classification is CSRL [4] (Conceptual Structures Representation Language) A significant portion of expert systems such as MYCIN [16] and PROSPECTOR [9] can be viewed as classification[2]

2. *Plan Selection and Refinement* is designing an object using hierarchical planning DSPL [2] (Design Specialists and Plans Language) is the tool for this generic task The task performed by the expert system MOLGEN [10] and R1 [12] can be viewed in this way

3 *Knowledge-Directed Information Passing* is determining the attribute of some datum based on the attributes of conceptually-related data The tool for this generic task is IDABLE (Intelligent DAta Base LanguagE) This task is often used in support of other tasks such as classification or design

4. *Hypothesis Matching* is matching hypotheses to a situation using an hierarchical representation of evidence abstractions The tool for this task is HYPER (HYPothesis matchER) Expert system PIP [19] can be viewed as performing this task at some stage in its reasoning

5. *Hypothesis Assembly* is constructing composite hypotheses in order to account for some set of data PEIRCE [13] is the tool for this task INTERNIST [1] and DENDRAL [3] systems largely perform this task.

As an example of the generic task concept, this paper will address the problem of diagnosis and how hierarchical classification solves some part of that problem Furthermore, we will discuss how the problem of sensor validation naturally integrates into a hierarchical classification of diagnosis In such an integration, a level of abstraction is provided that goes one step beyond traditional approaches that rely strictly on hardware redundancy

2. Classificatory Problem Solving

Diagnosis as a classification problem solving task is a matching of the data of the problem against a set of *malfunctions* (i e diseases, system failures etc) If the present data is classified as a known malfunction, then the diagnosis is completed. Note that this is a *compiled* approach to diagnosis as it requires that the possible malfunctions of the particular domain be pre-enumerated. Other less well defined problems require a deeper model that relies on first principles (physic, chemistry etc.) and

[1]Copyright (C) 1987 B Chandrasekaran and W F Punch III

[2]In fact, Clancey [7] has specifically analyzed MYCIN and shown it to be a kind of classification problem solving

an intimate understanding of the system at hand[3]

For example, given some data about a car engine problem, the hierarchical classification task is to find the categories (e.g., broken piston, faulty distributor) that best describe the data of the problem. The generic task characteristics of hierarchical classification are as follows.

2.1. Information Required and Results Yielded

Hierarchical classification requires as input a data description of the problem to be solved. After processing, this task yields all the categories of the malfunction hierarchy that apply to the given data.

2.2. Knowledge Required and its Organization

The classifier requires a pre–enumerated list of the categories that it will be using. Furthermore, these categories must be organized into a hierarchy in which the *children* (i.e. the subnodes) of a node represent subhypotheses of the *parent* (i.e. the superior node). Figure 1 illustrates a fragment of a tree from a hierarchical classification system for the diagnosis of Fuel System malfunctions in a car engine.

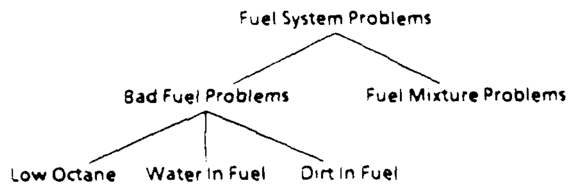


Figure 1: *Fragment of Fuel System classification tree*

Note that as the hierarchy is traversed from the top down, the categories (or in this particular case, hypotheses about the failure of the fuel system) become *more* specific. Thus the children of the hypothesis Bad Fuel Problems can be broken into more specific hypotheses of Low Octane, Water in Fuel and Dirt in Fuel.

Each node in the hierarchy is responsible for calculating the "degree of fit" or *confidence value* of the hypotheses that the node represents. For example, the Bad Fuel Problems node is responsible for determining if there is a bad fuel problem and the degree of confidence it has in that decision. Each node can be thought of as an expert in determining if the hypothesis it represents is present. For this reason, each node is termed a *specialist* in its small domain. To create each specialist, knowledge must be provided to make this confidence value decision. The general idea is that each specialist specifies a list of *features* that are important in determining whether the hypothesis it represents is present and a list of *patterns* that map combinations of features to confidence values. In the Fuel System Problems specialist, such features might include gas mileage problems, poor performance, difficulty in starting the engine etc. One pattern might be that if all the features are present, then the Fuel System Problems hypothesis is likely.

[3]Space and time limitations limit this paper to the compiled system issues; see reference [14] for a detailed discussion of the deep model issues and computational strategies.

2.3. The Control Strategy of Hierarchical Classification

Given that the knowledge of the system is organized as a set of specialists in a hierarchy, how can the hierarchy be efficiently traversed? This process is primarily accomplished through a type of hypothesis refinement called *establish–refine*. Simply put, a specialist that *establishes* its hypothesis (has a high confidence value) *refines* itself by activating its more detailed sub–specialists. A specialist that rules out or rejects its hypothesis (has a low confidence value) does not send any messages to its sub-specialists, thus avoiding that entire part of the hierarchy. The reason for this becomes obvious when one thinks again of how the specialists *are organized*. The subhypotheses of Fuel System Problems, for example, are simply more detailed hypotheses. If there is no evidence for Fuel System Problems (it is ruled out), then there is no point in examining more detailed hypothesis about failures of the fuel system.

The process of establish–refine continues until no more refinements can take place. This can occur either by reaching the tip level hypotheses of the hierarchy or by ruling out mid–hierarchy hypotheses.

3. CSRL, a Language Tool for Hierarchical Classification Systems

CSRL (Conceptual Structure Representation Language) is a language for writing hierarchical classification expert systems. As such, it allows a knowledge engineer to do three things:

1. Create a hierarchy of malfunction hypotheses in a particular domain.
2. Encode the pattern matching knowledge for each hypothesis into a specialist.
3. Control the process of establish–refine problem solving.

3.1. Encoding the Hierarchy of Malfunctions

In CSRL, a hierarchical classification system is implemented by individually defining a specialist for each malfunction hypothesis. The super– and sub–specialists of a specialist are declared within the definition. Figure 1 is a skeleton of a specialist definition for the Bad Fuel node from Figure 1. The declare section specifies its relationships to other specialists. The other sections of the specialist will be examined later.

```
(Specialist BadFuel
    (declare (superspecialist FuelSystem)
             (subspecialists LowOctane WaterInFuel
              DirtInFuel))
    (kgs ...)
    (messages ...))
```

Figure 2: Skeleton specialist for BadFuel

Designing a classification hierarchy is an important part of building a CSRL expert system, but the exact structure of the final system is a pragmatic decision rather than a search for the perfect hierarchy. The main criterion for evaluating a classification hierarchy is whether enough evidence is normally available to make confident decisions. To decompose a specialist into its subspecialists, the simplest method is to ask the domain expert what subhypotheses should be considered next. The subhypotheses should be subtypes of the specialist's hypothesis, and will usually differ from one another based on a single attribute (e.g., location, cause).

3.2. Encoding Pattern – Match Knowledge

The knowledge groups in the kgs section contain knowledge that matches the features of a specialist against the case data. Each knowledge group is used to determine a confidence value for some subset of features used by the specialist. As such, a knowledge group becomes an abstraction of evidence, representing an *evidential abstraction* of a particular set of features important to establishing the specialist. A knowledge group is implemented as a cluster of production rules that maps the values of a list of expressions (boolean and arithmetic operations on data, values of other knowledge groups) to some conclusion on a discrete, symbolic scale.

As an example, Figure 3 is the relevant knowledge group of the BadFuel specialist mentioned above. *It determines whether* the symptoms of the automobile are consistent with bad fuel problems. The expressions in the match part queries the user (who acts as the database for this case) concerning whether the car is slow to respond, starts hard, has knocking or pinging sounds, or has the problem when accelerating. AskYNU? is a LISP function which asks the user for a Y, N, or U (unknown) answer from the user, and translates the answer into T, F, or U, the values of CSRL's three-valued logic (Note that any LISP function may be used here). The results of the match expressions are then compared to a condition list in the with part of the knowledge group. For example, the first pattern "T ? ?" in the figure tests whether the first match expression (AskYNU? "Is the car slow to respond") is true (the ? means doesn't matter). If so, then −3[4] becomes the value of the knowledge group. Otherwise, subsequent patterns "? T ?" or "? ? T" are evaluated. The value of the knowledge group will be 1 if no rule matches. This knowledge group encodes the following matching knowledge.

> If the car is slow to respond or if the car starts hard, then BadFuel is not relevant in this case. Otherwise, if there are knocking or pinging sounds and if the problem occurs while accelerating, then BadFuel is highly relevant. In all other cases, BadFuel is only mildly relevant.

```
(relevant Table
  (match
     (AskYNU? "Is the car slow to respond")
     (AskYNU? "Does the car start hard")
     (And (AskYNU? "Do you hear knocking or
                    pinging sounds")
          (AskYNU? "Does the problem occur while
                    accelerating"))
    with (if T ? ?
           then -3
          elseif ? T ?
           then -3
          elseif ? ? T
           then 3
          else 1)))
```

Figure 3: relevant knowledge group of BadFuel

Figure 4 is the summary knowledge group of BadFuel. Its match expressions are the values of the relevant and gas knowledge group (the latter queries the user about the temporal relationship between the onset of the problem and when gas was last bought). In this case, if the value of the relevant knowledge group is 3 and the value of the gas knowledge group is greater then or equal to 0, then the value of the summary knowledge group (and consequently the confidence value of BadFuel) is 3 indicating that a bad fuel problem is very likely.

```
(summary Table
    (match relevant gas
      with (if 3 (GE 0)
             then 3
            elseif 1 (GE 0)
             then 2
            elseif ? (LT 0)
             then -3)))
```

Figure 4: summary knowledge group of BadFuel

This method of evidence combination allows the calculation of the confidence value to be hierarchically organized. That is, the results of any number of knowledge groups can be further abstracted by a knowledge group that can combine their values into a single confidence value.

3.3. Encoding of Establish – Refine Strategy

The messages section of a specialist contains a list of message procedures which specify how the specialist will respond to different messages from its superspecialist. Establish and Refine are the predefine messages in CSRL though others may be created by the user. The establish message procedure of a specialist determines the *confidence value* (i.e the degree of fit) of the specialist's hypothesis. Figure 5 illustrates the establish message procedure of the BadFuel specialist. relevant and summary are names of knowledge groups of BadFuel (see previous section). self is a keyword which refers to the name of the specialist. This procedure first tests the value of the relevant knowledge group. (If this knowledge group has not already been evaluated, it is automatically evaluated at this point.) If it is greater than or equal to 0, then BadFuel's confidence value is set to the value of the summary knowledge group, else it is set to the value of the relevant knowledge group. A value of +2 or +3 indicates that the specialist is established. In this case, the procedure corresponds to the following strategy.

> First perform a preliminary check to make sure that BadFuel is a relevant hypothesis to hold. If it is not (the relevant knowledge group is less than 0), then set BadFuel's confidence value to the degree of relevance. Otherwise, perform more complicated reasoning (the *summary knowledge group combines the values of other* knowledge groups) to determine BadFuel's confidence value.

```
(Establish (if (GE relevant 0)
             then (SetConfidence self summary)
            else (SetConfidence self relevant)))
```

Figure 5: Establish procedure of BadFuel

The refine message procedure determines what subspecialists should be invoked and the messages they are sent. Figure 6 shows a refine procedure which is a simplified version of the one that BadFuel uses. subspecialists is a keyword which refers to the subspecialists of the current specialist. The procedure calls each subspecialist with an establish message. If the subspecialist establishes itself (+? tests if the confidence value is +2 or +3), then it is sent a refine message.

[4] In this case, the values assigned are on a discrete scale from −3 to 3, −3 representing ruled-out and 3 representing confirmed.

```
(Refine (for specialist in subspecialists
            do (Call specialist with Establish)
               (if (+? specialist)
                 then (Call specialist
                        with Refine))))
```

Figure 6: Example refine procedure

4. The Computational Advantages of Hierarchical Classification

The major advantage of a hierarchical classification system is the organization of both the hierarchy of malfunctions and the knowledge groups within a specialist This organization allows an efficient examination of the knowledge of the system based on need

Consider again the hierarchy of Figure 1 The problem solving begins by evaluation of the specialist *Fuel System Problems* If that specialist establishes, then the two sub-specialists Bad Fuel Problems and Fuel Mixture Problems are invoked If however, the Bad Fuel Specialist does not establish, then none of its sub-specialists will be invoked Thus, if a specialist rules out (i e does not establish), then none of the knowledge of the sub-specialists need be run

The same is true of the knowledge groups in the specialist Only that knowledge necessary to confirm or deny the knowledge group is run If a row of the knowledge group matches, then none of the subsequent rows are evaluated Again, this results in running only the knowledge necessary for the problem at hand

Compare this with other so-called hierarchical approaches to diagnosis The *fault tree* is a sequence of causally related events that leads to an observable symptom in the system Given an initial malfunction, all possible causal results of the event are traced out, terminating with the symptoms that would be observed by a human diagnostician When applied to an entire system, the result is a network of events that represent all the causal relationships of the system's constituent parts While useful in design tasks, application of fault trees to diagnosis has a number of problems

1 The combinatorial fan-out from an initial event can be very large This makes the job of creating and traversing the network difficult Compare this with the abstraction of hypotheses in hierarchical classification systems Each node in the hierarchy represents a malfunction hypothesis that is listed in more detail through its sub-specialists If many sub-specialists occur in the hierarchical decomposition of the domain, more levels of abstraction can be introduced to limit the fan-out Such abstraction does not exist in fault tree representations

2 Fault trees make no attempt to limit the number of nodes of the network that must be evaluated Given a significant event, all possibilities are examined However, hierarchical classifiers make use of the abstraction of malfunction hypotheses to limit the number of nodes that must be examined based on the data of the case

5. Practical Applications of Hierarchical Classification

A number of diagnostic systems have been built using the hierarchical classification approach provided by the CSRL tool This section enumerates some the these applications and their domains

It should be noted that of the following systems Auto-Mech is strictly a pedagogical system, the Nuclear Power and Chemical Engineering systems are initial explorations for yet to be developed systems and Red, WELDEX and ROMAD are being developed to be used in real world situations

Auto-Mech [20]

Auto-Mech is an expert system which diagnoses fuel problems in automobile engines The purpose of the fuel system is to deliver a mixture of fuel and air to the air cylinders of the engine It can be divided into major subsystems (fuel delivery, air intake, carbuertor, vacuum manifold) which correspond to initial hypotheses about fuel system faults

Auto-Mech consists of 34 CSRL specialists in a hierarchy which varies from four to six levels deep Before running, Auto-Mech collects some initial data from the user This includes the major symptom that the user notices (such as stalling) and the situation when this occurs (e g , accelerating and cold engine temperature). Any additional questions are asked while Auto-Mech's specialists are running The diagnosis continues until the user is satisfied that the diagnosis is complete

A major part of Auto-Mech's development was determining the assumptions that would be made about the design of the automobile engine and the data that the program would use Different automobile engine designs have a significant effect on the hypotheses that are considered A carbureted engine, for example, will have a different set of problems than a fuel injected engine (the former can have a broken carburetor) The data was assumed to come from commonly available resources The variety of computer analysis information that is available to mechanics today was not considered in order to simplify building Auto-Mech.

Red [18]

Red is an expert system whose domain is red blood cell antibody identification An everyday problem that a blood bank contends with is the selection of units of *blood for transfusion during major* surgery The primary difficulty is that antibodies in the patient's blood may attack the transfused blood, rendering the new blood useless as well as presenting additional danger to the patient Thus identifying the patient's antibodies and selecting blood which will not react with them is a critical task for nearly all red blood transfusions

The Red expert system is composed of three major subsystems, one of which is implemented in CSRL The non-CSRL subsystems are a data base which maintains and answers questions *about reaction records (reactions of the patient's* blood in selected blood samples under a variety of conditions), and a overview system, which assembles a composite hypothesis of the antibodies that would best explain the reaction record. CSRL is used to implement specialists corresponding to the common blood antibodies and to each antibody subtype (different ways that the antibody can react)

The major function of the specialists is to rule out antibodies and their subtypes whenever possible, thus simplifying the job of the overview subsystem, and to assign confidence values, informing overview of which antibodies appear to be more plausible The specialists query the data base for information about the lab test results and other patient information, and also tell the data base to perform certain operations on reaction records.

Complex Mechanical Systems

CSRL has been used in creating expert systems that do diagnosis of faults both in the domain of Nuclear Power Plants and in the domain of Chemical Engineering

The Nuclear Power Industry must be very careful in the maintenance of running power plants since mistakes can prove costly not only in terms of power plant damage but also in terms of radiation leakage and broad environmental damage. Nuclear Power Plants are therefore heavily monitored in many areas, so heavily in fact that it is difficult (if not impossible) for the operator to maintain an understanding of just what exactly is going on. The Nuclear Power Plant expert system [11] is designed to take in large amounts of data and classify them into one of approximately 25 different failures. One advantage of the CSRL approach is that the operator can be informed of a high level view of the problem if no specific failure can be discovered.

The problems of the Chemical Engineering Plant are similar, but it does have a number of differences. While safety is also of concern, there is also the problem of product quality in a Chemical Engineering Plant. If a malfunction occurs that produces an unusable product, the operation must be brought quickly back into line or large amounts of material will be wasted. The Chemical Engineering expert system [17] does diagnosis of a typical reactor producing a solid product as a result of the reaction of liquid product and oxygen. It consists of approximately 30 specialists that represent hypotheses about failures of the various physical parts of the plant. In addition to data that monitors the state of the reactor, these specialists also use data about product quality to make the confidence value decision.

Other Real World Uses of CSRL

CSRL is being used to develop two commercial systems by the Knowledge Based Systems group at the Battelle Columbus Institute. WELDEX and ROMAD are diagnostic systems for, respectively, detecting welding defects and evaluating machinery. A brief description of WELDEX follows.

WELDEX identifies possible defects in a weld from radiographic data of the weld. Industry standards and regulations require careful inspection of the entire weld and a very high level of quality control. Thus for industries which rely on welding technology, such as the gas pipeline industry, radiograph inspection is a tedious, time-consuming, and expensive part of their operations.

This problem can be decomposed into two tasks: visual processing of the radiograph to extract relevant features of the weld, and mapping these visual features to the welding defects which give rise to them. WELDEX is intended to perform the second task. The current prototype consists of 25 CSRL specialists that are organized around different regions of the weld, taking advantage of the fact that each class of defects tends to occur in a particular region. The knowledge groups in these specialists concentrate on optical contrast, shape, size and location of the radiograph features. A customer version of WELDEX is currently being developed. Future work is anticipated on developing a visual processing system whose output would be processed by WELDEX, thus automating both parts of the radiograph inspection problem.

6. Data Validation in Hierarchical Classification

We have examined in some detail how hierarchical classification lends itself to performing diagnosis. However, there are a number of issues that exist in real-world systems that must yet be addressed. For example, most of the work in AI in diagnosis assumes the observations given to an expert system are reliable. In fact, it has always been understood that in real-world situations data can be unreliable and that human experts can still reach reliable diagnostic conclusions despite this unreliable data.

As an example, examine the domain of complex mechanical systems like Nuclear Power Plants or Chemical Processing Plants. Even before the advent of AI, system monitoring aids were designed that provided the operator with automatically validated data. These designs centered on monitoring each important system datum (pressure, temperature etc.) with a *number* of hardware sensors. This multiplicity of sensors provided a redundancy of information from which a more reliable value was extracted. Based on this hardware redundancy, a number of techniques were developed to validate a datum's value.

1. Providing more than one sensor of the same kind to monitor some datum. Loss of one sensor therefore does not preclude data gathering and any disagreements among the sensors can be resolved statistically for the overall datum value. For example, to measure the temperature of a chemical reaction, multiple temperature sensors could be used in the reactor and their statistical average given as the overall temperature value.

2. Providing different kinds of sensors to monitor a datum. This situation provides the same redundancy as 1) as well as minimizing the possibility of some kinds of *common fault problems*. That is, certain events that inactivate a particular sensor type may not affect sensors of a different type. Continuing with the example of 1) above, half of the sensors might be thermocouples while the other half might be mechanical temperature sensors.

3. *Using sensors in several different locations* to infer a datum value. In this situation, datum values are monitored not only directly but also inferred from other system data based on well-established relationships. For example, while the temperature of a closed vessel may be directly monitored, it can be inferred from the measurement of the pressure using the $PV = nrT$ equation.

While hardware redundancy allows some data validation, it has a number of drawbacks.

1. The expense of installing and maintaining multiple sensors for each important datum greatly increases the cost of the system.

2. Common fault failures still happen, especially as the result of severe operating failures.

3. Human operators and engineers resolve many such diagnostic problems using conflicting and even absent data. In other words, human experts are more tolerant of *bad* data whether it has been validated or not.

The following simple example will help in examining point 3) and other ideas[5]. Consider the mechanical system diagrammed in Figure 7 with data values indicated in Figure 8. It is a closed vessel with two subsystems, a cooling system and a pressure relief system. The vessel is a reactor which contains some process (nuclear fission, chemical reactions etc.) that produces both heat and pressure. The data values of Figure 8 indicate that the temperature of the reactor vessel, as determined by sensor hardware, is above acceptable limits. Assume that there are two possible causes of the high reactor temperature: either the *cooling system has failed or the pressure relief system has failed* and the added heat has overpowered the functioning cooling system. Given the sensor readings, what would be the diagnostic conclusion? The data conflict is the *normal* pressure and cooling system readings and the *abnormal* pressure relief system readings. The failure of the pressure relief system is plausible, data indicates its failure and no other system failure, but such a failure expects the pressure to be high! The step to take is to assume both the pressure relief system to have failed and the pressure sensor to be incorrect.

---

[5]Note that the ideas presented here have been used on more complicated real-world systems [8, 17, 15]. This example has been condensed from them for expository clarity.

The process shown above demonstrates data validation at a higher level than that of simple sensor hardware validation. In the example, the pressure system has failed despite the lack of a high pressure datum. However, there is other strong evidence[6] that the pressure system has indeed failed. The human reasoner *expects* the pressure datum to be high since the preponderance of other data indicate a malfunction. That is, the human reasoner in pursuing likely diagnostic conclusions discovers a plausible diagnostic conclusion that meets all but (in this case) one expectation. The important points to note are that

1. A diagnostic conclusion can and should be made based on the preponderance of other evidence
2. The datum value that does not meet expectation should be questioned and further investigation of its true value made

This is not to say that hardware redundancy is not useful for solving problems of conflicting sensor reports. The point is that hardware redundancy is made more useful in service of the higher level of redundancy provided by diagnostic expectations.

Also note that this process involves redundancy, not at the level of sensor hardware, but at the level of *diagnostic expectation*. This is a redundancy of information that allows questioning (and subsequent validation) of data based on multiple expectations of diagnostic conclusions. If a conclusion is likely, but not all of its expectations are met, then those now questionable values are investigated by more computationally expensive techniques.

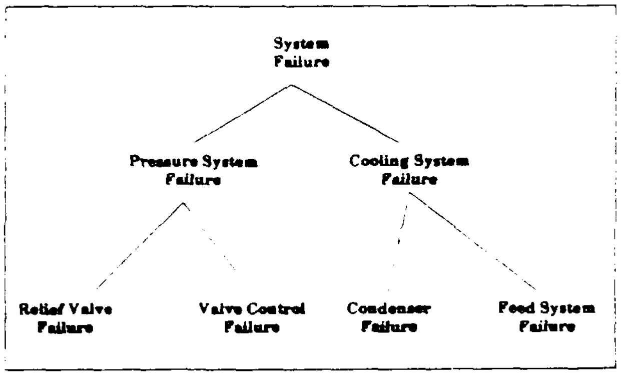


Figure 7: An Example Mechanical System

While space limitations restrict the scope of our discussion, the following section present the basic ideas concerning how the expectations of data can be naturally encoded in a hierarchical classification system. For further detail, please see [5].

6.1. Establishing a Questionable Datum

The process of establishing a questionable datum involves two steps. First, establish some expectations, using local knowledge or the context of other nodes. Second, use those expectations to flag some particular data value as questionable.

The expectations of a malfunction are embodied in the knowledge group. The knowledge group mechanism was designed to give a rating of pattern fit to data. If the fit is not perfect, those data values not meeting expectations are identified as questionable. In the example of Pressure Relief Valve Failure,

Variable	*Status*
Temperature	High
Pressure	Normal
Condenser	All sensors Normal
Cooling Water Flow System	All Sensors Normal
Relief Valve	Sensors indicate Malfunction
Valve Control System	All sensors Normal

Figure 8: Sensor Values for the Example

evidence exists that the valve has failed even though the pressure is normal. The lack of perfect fit between data and pattern allow the pressure value to be identified as questionable. Diagnosis continues despite apparent data conflict since enough evidence exists for establishing the malfunction hypothesis.

In establishing all the ancestors of any one node, a context of problem solving is created which provides a set of *implicit* expectations. Examine the example hierarchy of Figure 7. In order to establish the malfunction hypothesis *Relief Value* Failure, the malfunction hypothesis Pressure System Failure must have established. In the context of considering the Valve Failure, one can assume that some expectations were created based on the establishment of the Pressure System Failure node and other ancestors. Since these expectations always exist when considering Valve Failure i.e, you can't get to Valve Failure without establishing Pressure System Failure, they can be hard-coded into the Valve Failure Node.

How expectations are used for the Pressure Relief System Failure node is shown in Figure 9. A modification is made to the standard knowledge group first seen in Figures 3 and 4 that allows the expert to indicate both a match value for the group and a set of data that do not meet the expectations established at this stage of the problem solving. This is done by adding the keyword `Data-To-Question` to any pattern of the knowledge group, followed by a list of those data values that are considered questionable by the expert. Thus, Pressure Relief Failure establishes (based on other data features) despite the lack of a change of pressure. However, in establishing the node, one should question why the pressure did not change as expected. This is done by associating with the first pattern (the one that matches when the knowledge group establishes) the pressure datum by use of the `Data-To-Question` keyword. If that pattern is matched, then the match value is returned but the indicated data value is placed in a list of questionable data values will be later in detail. If the match value is high enough, the node establishes itself despite the existence of conflicting data. That is, if there is enough evidence to show a malfunction despite a conflicting value, the problem solving may continue. However, it may be the case that the value being questioned is of vital importance to establishing the node. The match value will reflect this, the node will not establish, but the data will still be placed on the questionable data list. After an initial run of the problem solver, the questionable data list will consist of datum values that did not meet the expectations of some node.

[6]In the example, this evidence is that there is a failure of the relief valve system which is part of the pressure system.

```
(pressure Table
 (match
   (AskYNU? "Is the Pressure Alarm on?")
   (AskYNU? "Is the Condenser Pressure High?")
   (AskYNU? "Is the Temperature Alarm
              activated?")

  with ((if F T T  Data-To-Question (pressure)
         then [illegible]
        [illegible]
         [illegible]
        [illegible]
```

Figure 9: Knowledge Group Modified for Data Validation

Once the set of questionable data has been identified, a number of methods may be applied to validate the value of the datum. These include

1. Examination of the values questioned (or not questioned) by other specialists in the hierarchy
2. Examination of the knowledge of sub-specialists to determine if the conflict can be resolved at that level
3. Use of more complicated and computationally expensive hardware checks that cannot be performed automatically on all data

7. Conclusion

Hierarchical classification is an approach to doing diagnosis for systems with compiled knowledge about the types of malfunctions that occur. It organizes this knowledge hierarchically and provides and efficient control strategy for examining those malfunction hypotheses based on the data of the problem at hand. Furthermore, the process of sensor validation has been shown to be a natural part of hierarchical classification. By adding a simple change to the *knowledge group*, data values that do not meet expectations can be identified and more powerful computational methods applied to resolve the conflict.

Acknowledgements

We gratefully acknowledge the support of grants from the Air Force Office of Scientific Research, AFOSR-82-0255, and the National Science Foundation, CPE-8400840. We would also like to thank Tom Bylander for help in preparing this document.

References

1. R A Miller, H E Popel, J D Meyers, INTERNIST-I, An Experimental Computer-Based Diagnostic Consultant for General Internal Medicine, *Readings In Medical Artificial Intelligence*, 1984, pp 190-209. First appeared in *New England Jrn of Med*, Vol 307

2. D C Brown, B Chandrasekaran, Knowledge and Control for a Mechanical Design Expert System, *IEEE Computer* 19, July 1986, 92-101

3. B Buchanan, G Sutherland, E A Feigenbaum, Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry, in *Machine Intelligence 4*, American Elsevier, New York, 1969, 1969

4. T C Bylander, S Mittal, CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling, *AI Magazine* 7(Summer 1986)

5. B Chandrasekaran, W F Punch III, Data Validation During the Classificatory Diagnostic Process, Submitted to AAAI87 for publication, 1987

6. B Chandrasekaran, Expert Systems: Matching Techniques to Tasks, *Artifical Intelligence in Business*, 1983, pp 116-132

7. W J Clancey, Heuristic Classification, *Artificial Intelligence* 27, 3 (1985), 289-350

8. J F Davis, W F Punch III, S K Shum, B Chandrasekaran, Application of Knowledge-Base Systems for the Diagnosis of Operating Problems, Presented to AIChe Annual Meeting, Chicago Ill, 1985

9. Duda, Richard O / Gaschnig, John G / Hart, Peter E, Model Design in the Prospector Concultant System for Mineral Exploration, In *Expert Systems in the Microelectronic Age*, Michie, D, Ed., Edinburgh University Press, 1980, pp 153-167

10. P Friedland, *Knowledge-based experiment design in molecular genetics*, Ph D Th., Stanford University, Computer Science Department, 1979

11. S Hashemi, B K Hajek, D W Miller, B Chandrasekaran, and J R Josephson, Expert Systems Application to Plant Diagnosis and Sensor Data Validation, *The Proceedings of the Sixth Power Plant Dynamics, Control and Testing Symposium*, Knoxville, Tennessee, April, 1986

12. J McDermott, R1: A Rule-Based Configurer of Computer Systems, *Artificial Intelligence 19, Issue 1*,(1982), 39-88

13. W F Punch III / M C Tanner / J J Josephson, Design Considerations for Peirce, a High Level Language for Hypothesis Assembly, *Expert Systems In Government Symposium*, October, 1986, pp 279-281

14. V Sembugamoorthy / B Chandrasekaran, Functional representation of devices and compilation of diagnostic problem solving systems, In *Experience, Memory and Reasoning*, J L Kolodner and C K Riesbeck, Eds., Erlbaum, 1986, pp 47-73

15. D D Sharma / D W Miller / B Chandrasekaran, Design of an Artifical Intelligence System for Safety Function Maintenance, *Transactions of the American Nuclear Society*, American Nuclear Society, November, 1985, pp 294-297

16. E H Shortliffe, *Computer-based Medical Consultations: MYCIN*, Elsevier/North-Holland Inc., 1976

17. S K Shum / J F Davis / W F Punch III / B Chandrasekaran, An Expert System for Diagnosing Process Plant Malfunctions, Presented to International Federation of Automatic Controls, Kyoto Japan, 1986.

18. Smith, Jack W / Svirbely, John R / Evans, Charles A / Strohm, Pat / Josephson, John R / Tanner, Mike RED A Red-Cell Antibody Identification Expert Module *Proceedings of the Eighteenth Annual Hawaii International Conference on System Sciences*, The Ohio State University, 1985, pp 126 – 135

19. P Szolvits / S G Pauker Categorical and Probabilistic Reasoning in Medical Diagnosis *Readings in Medical Artificial Intelligence*, 1984, pp 210 – 240 First appeared in *Artifical Intelligence*, Vol 11

20. Tanner, Michael C / Bylander, Tom Application of the CSRL Language to the Design of Expert Diagnosis Systems The Auto-Mech Experience Technical Report, The Ohio State University Dept of Computer Science November 12, 1983.

AFOSR-TR- 88-1600

AIAA-87-1682
AI Applications for Space Support and Satellite Autonomy

C. J. Golden, Ford Aerospace &
Communications Corp.,
Sunnyvale, CA

Second AIAA/NASA/USAF Symposium on
Automation, Robotics and Advanced Computing
for the National Space Program
March 9-11, 1987/Arlington, VA

AI APPLICATIONS FOR SPACE SUPPORT AND SATELLITE AUTONOMY

Constance J. Golden*
Ford Aerospace & Communications Corp.
Sunnyvale, CA

ABSTRACT

Knowledge based systems (KBS) have not yet met expectations. Situations that haven't been anticipated or planned for must be handled by KBSs if they are to be useful for complex space support applications. This requirement implies that "deep knowledge" about the domain must be modeled so the KBS can "reason" about the situation and generate and implement a plan for recovery or action. Domains may be represented using several schemes, each having limitations that prevent them from meeting expectations for KBS's. Ford Aerospace has developed a representation approach that is a hybrid of other schemes. Because this approach represents knowledge in a complete, consistent and unambiguous manner, generic approaches to processing domain information can be developed, and unplanned situations can be analyzed and "reasoned" about. This approach promises to meet most, and possibly all, of the space support requirements for KBSs. Ford Aerospace's KBS architecture has been applied to several space support activities in an effort to test its full capabilities in being able to "reason about" and "analyze" new situations. Satellite autonomy, equipment maintenance and network control applications are among those discussed.

I. Space Support Application Requirements

Knowledge Based Systems (KBS) have been successfully applied to automate some space support functions, but the expectations have usually exceeded the implemented capability. In some areas of satellite control it is acknowledged that no existing KBS could replace the expert facing an unforeseen event. For example, a solar storm caused a bit change in the solar array adapter control of a satellite which locked the solar array tracking mechanism. (see Figure 1.) As the satellite progressed along its orbit, the solar energy impinging on the panels decreased because the panel could not maintain its normal perpendicular position to the sun's rays. Eventually the power from the solar array was so low that the vehicle automatically shed its power load and turned off the attitude control subsystem. The tumbling vehicle was detected and placed in a safe, sun pointing, spunup configuration. Analysis of telemetry data showed the cause of the locked solar array to be a spurious bit change. This bit change was corrected and the procedure for returning to a stable earth pointing attitude was started. The approved procedure, when implemented, failed to result in the expected earth pointing attitude. Factory experts were consulted and it was decided that the thrusters used in the maneuver impinged on the solar array panels, imparting forces that prevented normal earth acquisition. A second earth acquisition maneuver was planned using a different set of thrusters, known to be directed away from the solar panel as oriented. This procedure was a success and the satellite was returned to its functional earth pointing attitude.

Another example involves an R&D satellite which lost earth lock shortly after initial earth pointing attitude control had been established. There was no apparent reason (e.g., sun/moon interference, eclipse, power decrease, etc.) for the problem. A close look at the telemetry data, just prior to start of vehicle tumbling, showed that the earth sensor data being used by the on-board software was not the expected data. Further research revealed that the installed earth sensor design characteristics were different from those written in the control software. A corrected version of the control software was uploaded to the satellite. The earth acquisition procedure was repeated and the attitude control system maintained earthlock. While we may not expect KBSs to automatically correct the on-board software we do expect KBSs to diagnose the problem immediately upon receipt of the first unexpected earth sensor data and recommend a solution to the operator. This requirement implies that orbital (flight) information must be available and "deep knowledge" about the satellite must be modeled so that the KBS can "reason" about the situation and generate and implement a plan for recovery or action.

*Manager, Advanced Programs
Senior Member AIAA

II. Alternative KBS Approaches

Domains may be represented using rules, frames, networks or other schemes. Some of the more common representation alternatives are compared on Table 1. They all fall short of meeting the expectations listed in Figure 2 for KBSs in space support applications. For example, generic processing is provided by the inference engine in rule-based systems, but this processing is limited to pattern matching between situational data and data referenced within rules. To allow for more powerful generic processing, the knowledge must have a better defined structure than that currently used for rules. The structure itself includes information that can then be used by processing modules to perform more specific functions than pattern matching. In addition, the flexibility (or lack of structure) of rule-based systems, makes system Verification and Validation (V&V) nearly impossible because it is difficult to isolate a part of the system for test: Any rule that fires may generate unexpected side effects within other parts of the system.

For a KBS to reason about or adapt to events or situations for which it was not preprogrammed, deep or causal knowledge about the domain must be represented in an explicit manner that can be accessed by the reasoning modules. Knowledge represented within rules or methods is only used during actual execution, and only the result of the execution is available for reasoning; the information contained within rules or methods is inaccessible to a problem solving module.

Major inadequacies of all existing systems meant to be used for real space support applications are presented on Figure 2 and include: the need for AI expertise during system development, the inability to handle situations which haven't been explicitly entered into the knowledge base, the lack of a formal V&V approach, and the inability to provide real-time solutions to a complex situation.

Ford Aerospace has developed a KBS approach, called PARAGON, that is a hybrid of other approaches. It promises to meet most, if not all, desired characteristics of a space support KBS. The domain is modeled using a semantic network approach where the concepts (e.g., physical or non-physical objects), appropriate characteristics of the concepts (e.g., height, weight, color), interaction or relationships with other domain concepts (e.g., electrically connected, heat supplied by, etc.), and the behavior of the concept (e.g., states it may be in such as on, off, or idle; what events occur while in each state;

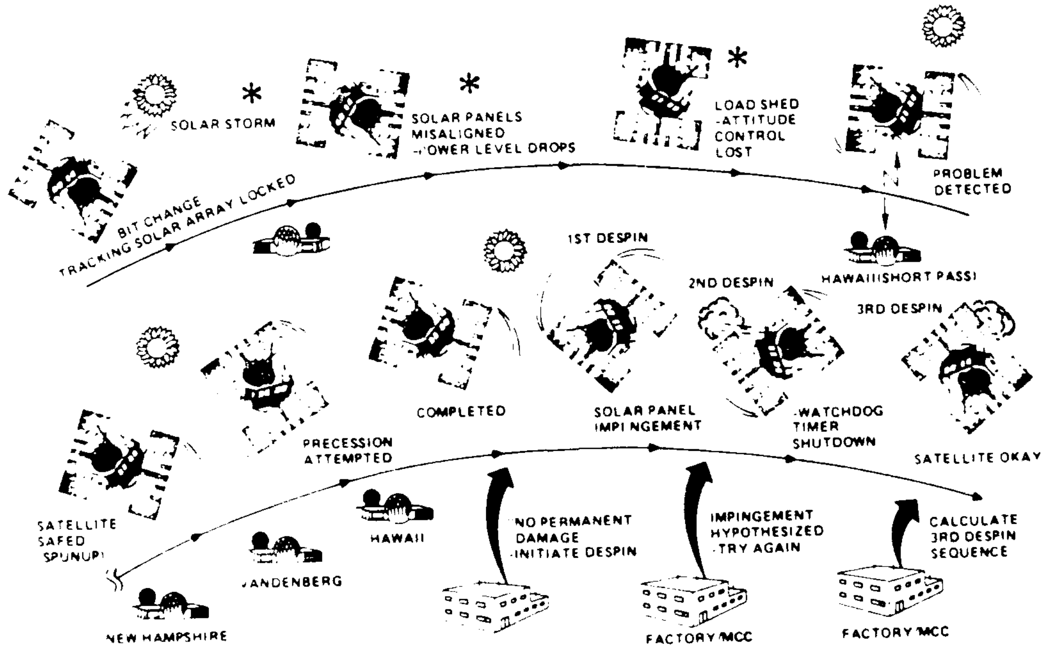


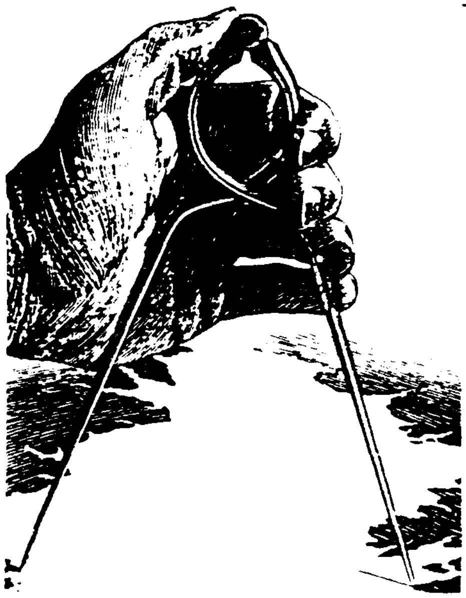
Figure 1. Example - Satellite Anomaly

Table 1. Comparison of Representation Alternatives

REPRESENTATION	DESCRIPTION	STRENGTHS	WEAKNESSES
[illegible]	SITUATION ACTION PAIRS	*FLEXIBLE *STAND ALONE *REPRESENT POORLY STRUCTURED INFORMATION *AVAILABLE DEVELOPMENT TOOLS	[illegible]
[illegible]	FRAMES OF RELATED FACTS AND BEHAVIOR USING MESSAGE PASSING FOR BEHAVIOR	*FLEXIBLE *DATA AND BEHAVIOR PACKED TOGETHER *MAINTENANCE *AVAILABLE DEVELOPMENT TOOLS	[illegible] METHODOLOGY
METHODS	DESCRIPTION OF BEHAVIOR	*ALLOW USE OF BEHAVIOR FOR ACTION IN SYSTEM	*BLACK BOX THAT CANNOT BE USED FOR REASONING
CLASSIFICATION HIERARCHY	HIERARCHICALLY ORDERED FRAMES OF RELATED FACTS AND BEHAVIORS	*STRUCTURE *INHERITANCE *MAINTAINABLE	*LACK OF CONSISTENT DEFINITION BETWEEN LEVELS *LACK OF DEVELOPMENT METHODOLOGY *ONE DIMENSIONAL LIMITS NUMBER OF RELATIONSHIPS REPRESENTED
DEMONS	CONDITIONALLY ACTIVATED BEHAVIORS	*EFFICIENTLY MONITORING FOR RELEVANT EVENTS	*DIFFICULT TO ORGANIZE AND CONTROL
BLACKBOARDS	COMMUNICATION STRUCTURE THAT ALLOWS INTERACTION BETWEEN MULTIPLE KNOWLEDGE SOURCES AT VARYING LEVELS OF DOMAIN	*MULTIPLE LEVELS *ABILITY TO DEFINE INTERACTION BETWEEN LEVELS *INDEPENDENT KNOWLEDGE SOURCES CONTRIBUTE	*COMPLEXITY OF DEFINITION *LACK OF EXPLICIT CONTROL METHODOLOGY *DIFFICULT TO USE WITH TIME DEPENDENT KNOWLEDGE
SEMANTIC NETWORKS	GRAPH NODES (REPRESENTING CONCEPTS) AND LINKS (REPRESENTING RELATIONSHIPS)	*WIDE VARIETY OF RELATIONSHIPS REPRESENTED *SOME METHODOLOGY *NATURAL REPRESENTATION	*AMBIGUOUS DEFINITION OF RELATIONSHIPS ALLOWED *LACK OF DEFINED PROBLEM SOLVING METHODS
FIRST ORDER LOGIC	COMPLETE MATHEMATICAL DESCRIPTION OF FACTS	*COMPLETENESS *COMPATIBLE	*COMPLETENESS DIFFICULT TO ACHIEVE *DIFFICULT TO REPRESENT COMPLEX PROCEDURAL CONTROL

CURRENT CAPABILITY

- KNOWLEDGE SEPARATED FROM PROCESSING
- LIMITED GENERIC PROCESSING
- REFLECTS EXPERTS EXPERIENCE ONLY
- SYSTEM DEVELOPMENT REQUIRES AI EXPERTISE
- DIFFICULT TO V&V
- NON-REAL TIME PROCESSING
- RELATIVELY EASY TO MAINTAIN



DESIRED CAPABILITY

- POWERFUL GENERIC PROCESSING
- LEARNS FROM EXPERIENCE
- HANDLES NEW SITUATIONS
- ADAPTS TO CHANGING ENVIRONMENTS
- DOMAIN EXPERT BUILDS AND TESTS SYSTEM
- FORMAL V&V POSSIBLE
- REAL TIME PROCESSING

Figure 2. Knowledge Based Systems - The Expectation Gap

and what causes transition from one state to another) are described via a graphic interface and menus. Typing can be limited to assigning names to concepts, states, etc.

As the model is being developed, PARAGON collects the information and automatically translates it to a machine representation useful for inferencing and problem solving. Characteristics, relationships and behavior must be defined precisely and within constraints to avoid the problem of ambiguous definition found in most semantic networks and to allow formal V&V. The approach allows PARAGON to automatically generate LISP code so a simulator for the domain is available to test behavior and display characteristics for verification by the system developer. Some minor modifications would allow generation of other types of code such as C, PASCAL, FORTRAN, etc., so the resultant model of the knowledge base could be used as a stand-alone simulator running on a non-LISP machine.

During development of this portion of PARAGON, two guiding principles were followed: that the model be declarative and consistent. We wanted the model to be declarative so all control knowledge (or algorithms for processing knowledge) would be eliminated from the knowledge base and all knowledge describing each concept would be internal to the data structure itself. Semantic networks allow representation of a domain with no control or processing of knowledge required. Also the representation elements are easy for engineers to work with. Therefore this was our starting point. However, any type of relationship is allowed in a "pure" semantic network which can lead to inconsistencies. Therefore, constraints had to be imposed on the way concepts and relationships are defined and the way relationships interact to ensure consistency. Other aspects of representation schemes such as frames, rules and blackboard were used to constrain the knowledge base representation. Since many different systems and users may want to access the same knowledge base for a variety of purposes and since two or more knowledge bases may have to interact to find a solution for a user, consistency was absolutely an essential requirement for space support applications.

Since the domain model in PARAGON contains no control or processing knowledge, reasoning modules have been and are being developed to perform their function on the domain model, independent of specific domain information. Cognitive psychologists, Artificial Intelligence (AI) researchers and knowledge engineers have identified the following most commonly used processes or strategies:

1. Monitoring (data interpretation): comparing observations to planned (desired) situation

2. Interpretation (situation assessment): inferring situation description from sensor data

3. Diagnosis: inferring cause of system malfunctions from observables

4. Debugging: prescribing remedies for malfunctions

5. Prediction: inferring likely consequences of given situations/actions

6. Design: configuring or modifying objects under constraints, such as available resources

7. Goal Determination: debugging, prediction, design and resource allocation (knowing desired situation)

8. Planning: designing actions to achieve goal

9. Repair (corrective action): executing a plan to administer a prescribed remedy

10. Instruction: diagnosing, debugging and repairing student behavior

The reasoning process followed for each of the basic strategies above is programmed into a separate module. The builder of an expert system takes the model of the domain, adds the reasoning modules appropriate for the application and then builds the operator/system interface needed with the outside world to run the system.

Figure 3 illustrates the overall development process.

III. Space Support Applications

Ford Aerospace's knowledge-based system architecture has been applied to automate several functions in an effort to test its full capabilities in being able to "reason about" and "analyze" new situations. Illustrative examples and lessons learned are presented below:

Satellite Autonomy: The first step in achieving satellite autonomy is to automate the satellite control function on the ground. In many ways this step is more difficult than the final step in space, because the ground station doesn't

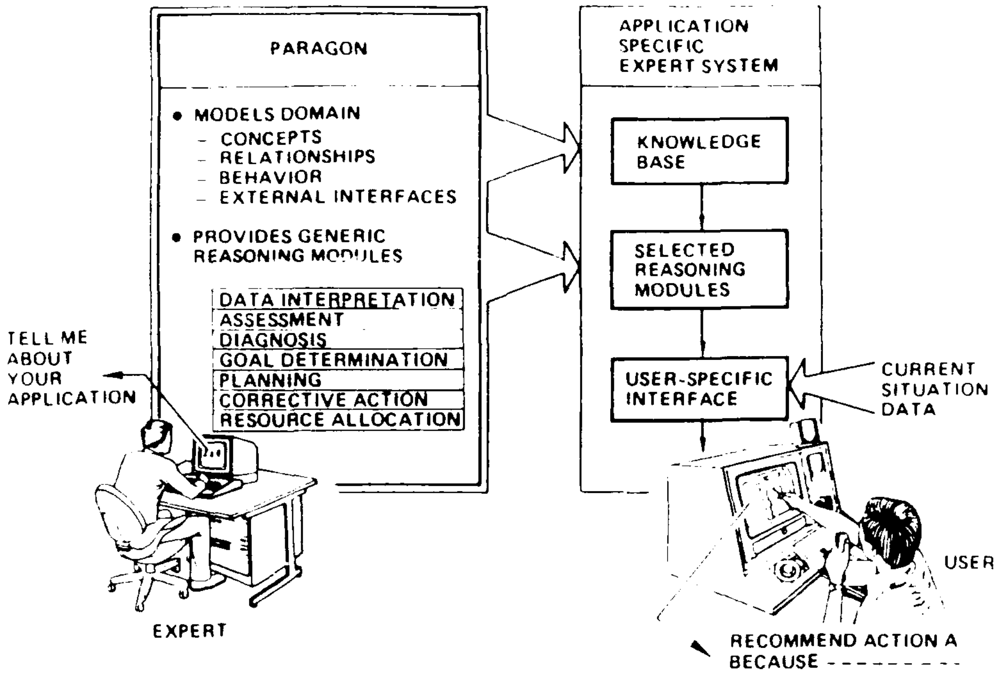


Figure 3. PARAGON - A Generic Expert System Development Environment

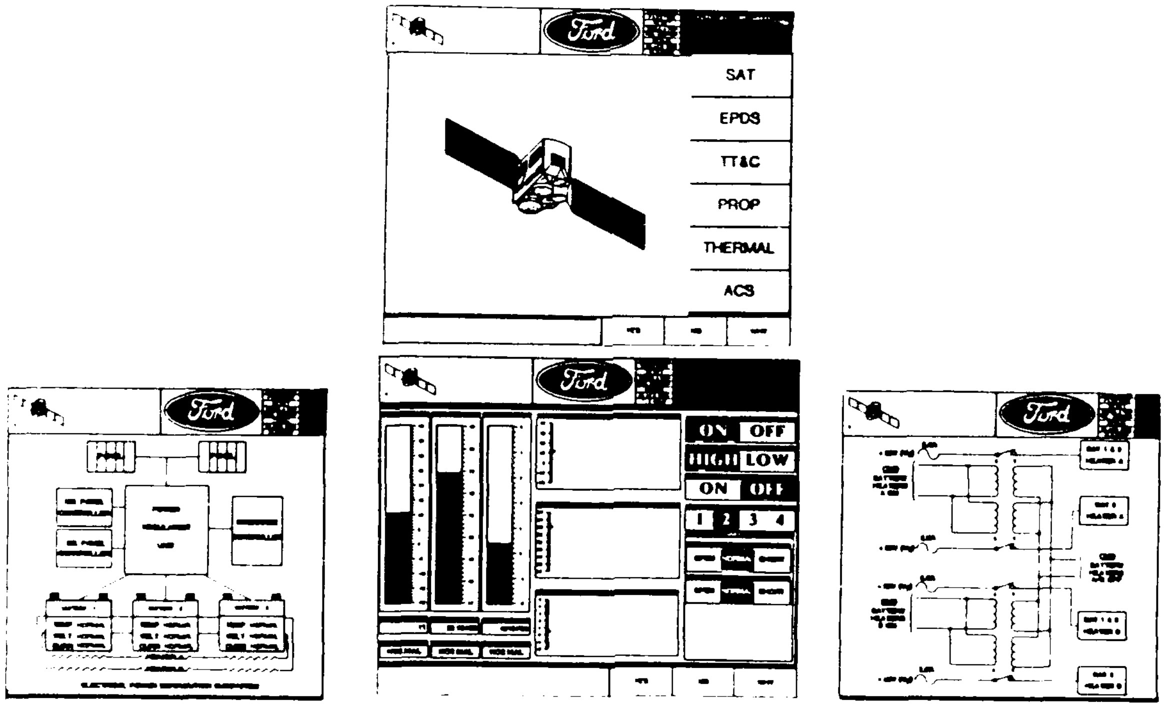


Figure 4. STARPLAN Display

always have continuous contact with the satellite and hence must reason with incomplete knowledge about what happened between passes. Our version of an automated satellite control system is called STARPLAN. Figure 4 illustrates some of the user interface screens. It has been developed using several architectures: all rule-based, frame-based and PARAGON representations. To date GPS and DSCS subsystems have been modeled and known anomalous situations have been presented as input and handled by the system. Based on our experience, there are three areas where further work is necessary at this time on this application.

1. The satellite design engineers find that time critical feedback loops can only be represented accurately if precise definitions of the concepts involved and their relationships are provided. These are needed to take full advantage of the simulation capability within PARAGON. We are considering provision of a set of preprogrammed concepts and relationships for this situation so the user can choose the one that fits his design characteristics most closely. This approach to aiding the design engineer is non-trivial, because each feedback loop is inter-related to other unique domain knowledge being entered by the engineer.

2. Our generic causal diagnosis processing module sometimes stops at the wrong level in finding the cause of a problem. I'll use an example to explain what I mean. When you get in your car and turn the key, you expect the engine to start. If it doesn't, you note that this information deviates from what was expected and reason about possible causes for the discrepancy. A likely cause is a dead battery, so you turn on the light switch and note that the lights don't work. The result of this test makes a dead battery even more likely as a cause, so you put in a new battery. What if the engine still doesn't start, or it starts for a short time and then repeats the anomalous behavior? The cause could be a short in the system which causes the battery to go dead or the engine and lights to be inoperable. For an automobile, there are approaches that can be taken to test for malfunctions that would be at the next level beyond an apparent cause. Also, since batteries don't cost much and the risk is low, because the system probably won't be harmed if a new battery is installed, the KBS should make this recommendation and only look for another level of malfunction if that recommendation fails. However, satellites operate in a completely different environment. The most likely cause of a discrepancy shouldn't be fixed with a replacement until the risks and next level of causes have been evaluated. Therefore, our causal diagnosis, goal determination, planning and corrective action modules must all work together iteratively to provide the most appropriate correction recommendation for an orbiting satellite. Different domains will require different levels of cause identification. How best to incorporate this information is still being investigated.

3. Real-time operation is still to be demonstrated. Until reasoning issues and domain expert interface issues are resolved, it doesn't make sense to concentrate on real-time considerations. Also, parallel hardware systems are becoming more powerful every day. This medium will be a major factor in achieving real-time operations.

<u>Remote Maintenance of Electronic Equipment</u>: The systems we have demonstrated have used distributed knowledge bases, since the information available at the remote location is restricted to the site, while the information at the central control center includes global information that impacts priorities and constrains corrective action alternatives.

<u>Digital Design</u>: A system that performs fault isolation on a digital board has been prototyped. Chips and the wires connecting them were modeled as well as individual behavior (Note: wires have individual behavior patterns such as propagation time, delay time, resistance, etc.). The major problem encountered was timing, similar to the feedback loop timing problem discussed under satellite autonomy.

<u>Network Control</u>: A system that models a communication satellite with its individual transponders and the ground stations that make up its network has been prototyped. If a communication link isn't working as planned, the system decides whether weather, jamming, loading or equipment failure is the cause and then recommends a solution. If weather is the cause, using an alternative ground station and a connecting ground link might be the recommended solution. Time synchronization is a consideration in this application. PARAGON currently models timing as a continuous behavior. Further enhancements to the development interface will make it easier to describe delay times and cyclic processes.

<u>Manufacturing Plant Process Control</u>: PARAGON is being used by Ford Motor Company to model one of their manufacturing plants and simulate the process (from the top down). Used only as a simulation tool, the sensitivity of material in process and inventory to schedule can be

evaluated. Once the "reasoning" modules are added, inefficient situations can be detected, the cause of the problem identified and recommendations for improvement made. At first we considered using existing domain simulators that represent many years of development effort, but the knowledge is not appropriately represented for a reasoning module to use. Translation may be feasible for some small, simple domains, but not for large, complex domains. The lesson learned is: build the domain model using the PARAGON representation scheme, test and V&V the knowledge base using the model as a simulator, add generic (or domain specific if necessary) reasoning modules that have been tested and V&Vd separately, build an interface with the external world (operator, telemetry stream, etc.) and you have a powerful simulator within an even more powerful KBS that will meet most of the desired space support requirements for KBSs.

PARAGON currently is being used in an R&D and staff consultant environment. Extensions need to be made to the system to increase its reasoning capability. Real-time applications will be pursued to test its parallel reasoning structure on parallel processing hardware. Maybe next year we can demonstrate an even further narrowing of the KBS expectation gap!

AFOSR-TR- [illegible]

AIAA-87-1685
Validation of Knowledge-Based Systems

R. A. Stachowitz, J. B. Combs and
C. L. Chang, Lockheed Missiles &
Space Company, Inc.,
Austin, TX

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program
March 9-11, 1987/Arlington, VA

Validation of Knowledge-Based Systems

Rolf A. Stachowitz, Jacqueline B. Combs and Chin L. Chang
Lockheed Missiles & Space Company, Inc.
AI Center - Austin, O 90-06, B/30E
Austin, Texas

Abstract

Work on providing a comprensive validation capability for knowledge-based systems has been progressing successfully at the Lockheed AI Center--Austin for more than a year. The Expert Systems Validation Associate (EVA) already extends and complements the validation capability provided by existing commercial or academic expert system shells. In this paper we outline the architecture, theoretical basis, and functionality of EVA and describe its existing and future features. The existing features are currently implemented for knowledge-based systems developed in ART (Automated Reasoning Tool, Inference Corporation).

INTRODUCTION

Knowledge-based system (KBS) technology has now emerged from applied artificial intelligence (AI) as a vital technology for modeling complex systems in defense, industry, business, and science.

The growing reliance on KBSs requires the development of appropriate methods for validating such systems. The objective of the Expert System Validation Associate (EVA) -- under development at the Lockheed AI Center at Austin, TX -- is to define and develop automated tools to validate the structural*, logical, and semantic integrity of KBSs. In this paper, which is a revised and expanded version of a previous paper[1], we define KBS validation as the verification, validation, and testing of a KBS. Our definition does not include the evaluation of the quality of a system by a group of experts.

Literature on validating traditional software abounds (for a condensed bibliography, see Miller[2]). The validation of KBSs, however, has received considerably less attention in the AI literature. Exceptions are Nguyen et al.[3] and Suwa et al.[4].

Validation methods for traditional software are difficult, time-consuming, and not necessarily correct. We have learned to use the expression *correct with respect to specifications* rather than plain, unrestricted *correct.* Traditional validation -- represented by the *waterfall model*[5] -- involves the complete software development life-cycle and is not oriented towards the incremental development method typical of knowledge-based systems.

During the construction of typical KBSs, requirements are most of the time not as precise as the requirements for a conventional software development project. Thus the system designer needs validation tools to assist him/her in clarifying the requirements and obtaining a precise specification. This is typically obtained through rapid prototyping of a partial, incomplete specification. Based on the output obtained, the specification is modified (corrected, deleted, or extended), until the problem appears to be sufficiently well formulated and understood. An appropriate model for the development of such systems, the *spiral model*, has recently been developed by Boehm[6].

A second reason for the difficulty of applying conventional validation methodology to a KBS is that a KBS does not exhibit the functional characteristic of standard procedural software modules where given input(s) are mapped into specified output(s). A KBS is mostly nonprocedural, and it normally does not contain components analogous to standard software modules amenable to validation by conventional tools.

It is our conjecture that software validation can be more easily performed in a KBS environment. In such an environment the number of life cycle steps is reduced from the traditional four (requirements development, specification development, design development, code development) to just the first two, resulting in a considerable reduction in the amount of validation work to be performed. In addition, KBSs can be regarded as logic-based systems, consisting of a declarative (application) and control component (inference engine). The logic-based languages used to implement the applications are more amenable to validation using logic and mechanical theorem proving. Finally, the availability of metaknowledge with semantic information and integrity constraints permits (a) the formulation of conditions (knowledge base designer's specifications) that the rules and facts of a KBS have to meet and (b)

*pertaining to "syntactic" properties of rules, clauses, and facts in a knowledge base the automatic checking of the validity of the underlying application.

KBS DEVELOPMENT AND VALIDATION

The incremental development process used in building KBSs is represented in figure 1, adapted from Boehm's spiral model.

The spiral begins with a set of initial requirements, describing the problem to be solved. The requirements phase is followed by the specification phase, in which the expert's knowledge about the domain in question is translated into rules and facts. Because of the clear separation between declarative and control component, these specifications can be executed immediately, resulting in an *initial operational capability* (IOC), a first prototype. The expert or knowledge engineer repeats this process (goes through the spiral), expanding or revising the requirements and specifications until after many iterations (with many *interim operational capabilities*) the *final operational capability* (FOC), the end product, has been obtained.

It should be obvious that the number of iterations could be reduced considerably given the availability of an automated validation tool.

EVA GOAL

The goal of EVA is to provide automated tools to validate the logical, structural, and semantic integrity of KBSs. In addition, EVA also addresses the goal of improving the validation process by finding mistakes and omissions in the knowledge base, proposing knowledge base extensions, restrictions, and generalizations. In other words, EVA addresses not only the question of "Is a KBS application correct?" but also the question of "Is the knowledge used by the expert for the application correct?"

EVA ARCHITECTURE

To permit the addition of future functionality EVA is designed to be a *meta*knowledge-based system shell whose architecture is shown in Figure 2.
Standard commercial expert system shells, also called *object* shells, provide the following basic constructs: objects, classes of objects (object schemas), generalization hierarchies among the classes of objects, attributes (slots) of objects, property inheritance, relation schemas, facts, rules, and so on. Applications are written in the object shells by application programmers or knowledge engineers.

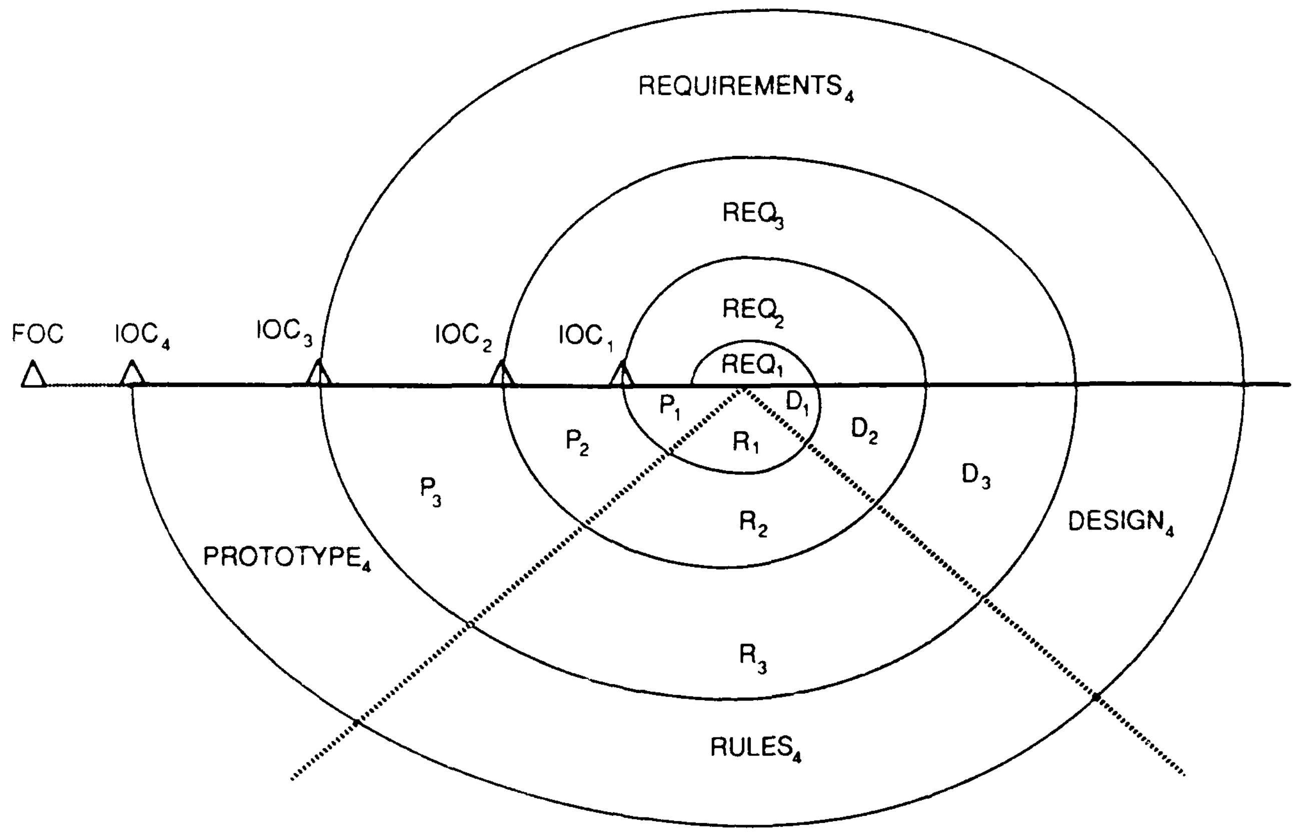


Figure 1. Boehm's spiral model

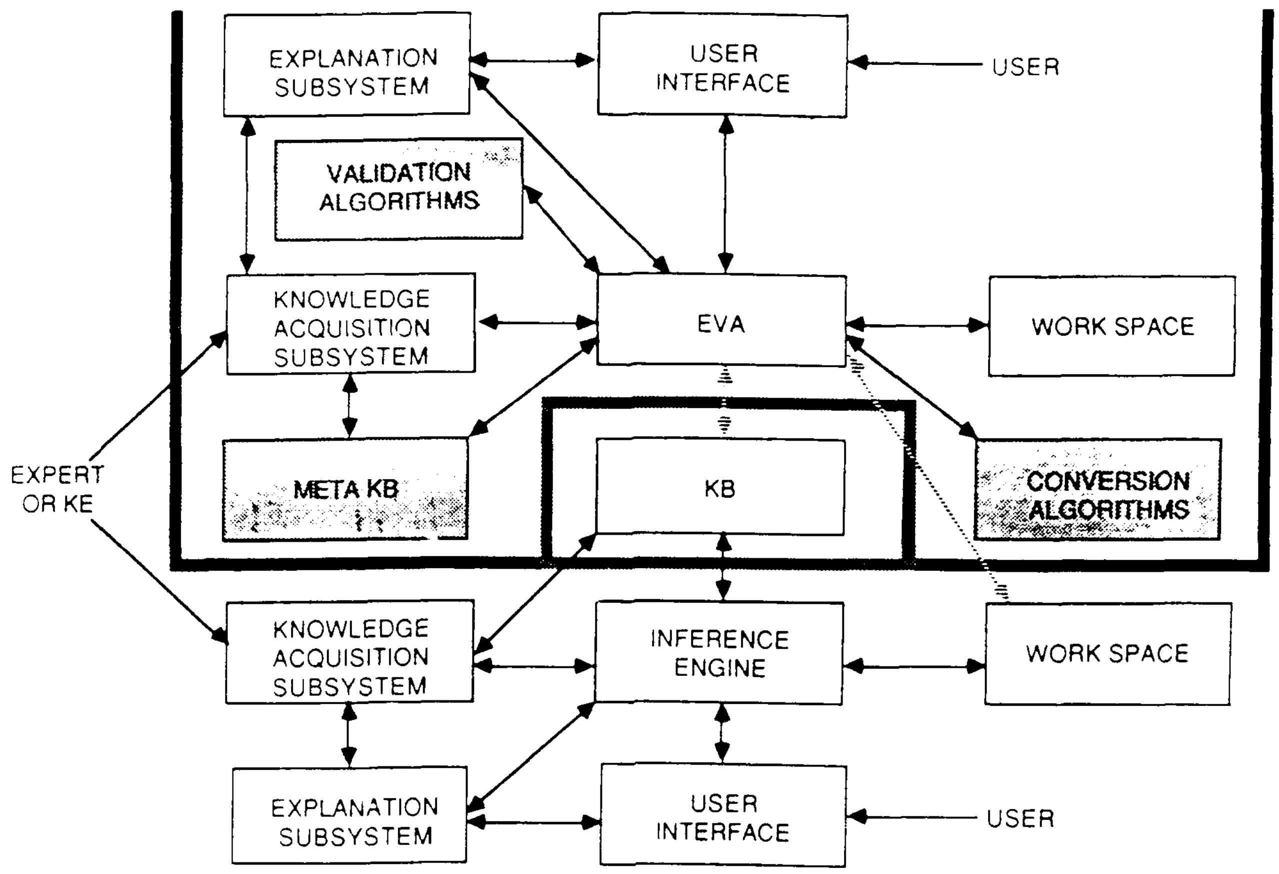


Figure 2. EVA architecture

In contrast to the knowledge base of an object shell, EVA provides a metashell with a metaknowledge base that supports higher order constructs to represent knowledge, properties, and constraints about the lower level objects, object schemas, relation schemas, the ordering of rules, and so on, of an application.

EVA FUNCTIONALITY

The purpose of our research effort is to design and implement a very expressive metashell based upon higher-order constructs. For example, it can be used to represent that a relation is symmetric, nonsymmetric, asymmetric, transitive, nontransitive, intransitive, reflexive, irreflexive, connected, mandatory, critical, synonymous to some other relation, and others. The metashell can be used to represent validation statements on constraints and control strategies for the application by an application system designer.

The functionality of EVA, represented by means of a data flow diagram, is depicted in Figure 3.

EVA contains two kinds of functional components: modules specific to a particular object shell (i.e., conversion algorithms used by the analyzer) and modules specific to EVA, the validation tools.

The analyzer uses conversion algorithms to translate the application (represented by rules, facts, and schemas of the object shell) and the validation statements into the EVA format to be used by the validation modules.

These validation modules are the logic checker, structure checker, semantics checker, omission checker, rule proposer, behavior verifier, and control checker. The logic checker, structure checker, and behavior verifier address the question "Is the application written in the object shell correct?", while the semantics checker, omission checker, rule proposer, and control checker address the question "Is the knowledge used by the expert (knowledge engineer) for the application correct?" For example, the sentence "An automobile does not need motor oil" is grammatically correct, but it conveys incorrect information.

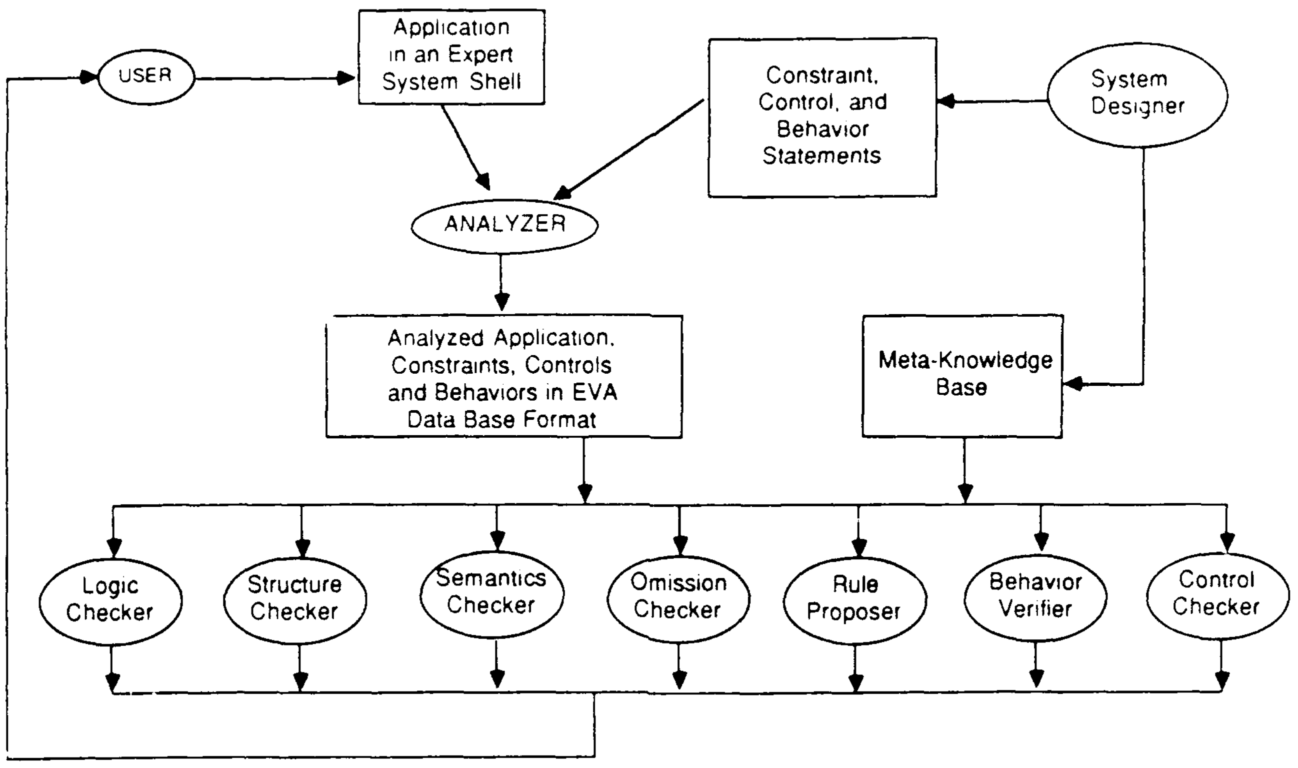

USER
Application in an Expert System Shell
Constraint, Control, and Behavior Statements
System Designer
ANALYZER
Analyzed Application, Constraints, Controls and Behaviors in EVA Data Base Format
Meta-Knowledge Base
Logic Checker
Structure Checker
Semantics Checker
Omission Checker
Rule Proposer
Behavior Verifier
Control Checker

Figure 3. EVA functionality

We have implemented considerable portions of the EVA functionality for ART-based[*] expert systems. Our efforts, so far, have concentrated on the implementations of the logic checker, the structure checker and the semantics checker.

LOGIC CHECKER

The purpose of the logic checker is to check whether the rule base of an application is consistent and "numerically complete."

Consistency

The logic checker checks for two types of inconsistency: direct and indirect. Two rules are directly inconsistent if they contain mutually exclusive right-hand sides (RHSs) and the left-hand sides (LHSs) are equivalent (identical, or one LHS subsumes the other). Indirect inconsistency occurs when two rules whose LHSs are equivalent derive contradictory RHSs after rewriting intermediate RHSs.

In the following examples, rules 1 and 2, and rules 3, 4, and 5 are inconsistent and are reported by EVA.

[*] Automated Reasoning Tool, Inference Corporation

Direct Inconsistency

Rule 1: STUDENT(x)∧STAFF(x)
$\rightarrow$UNIVERSITY_MEMBER(x)

Rule 2: STUDENT(x)$\rightarrow \neg$UNIVERSITY_MEMBER(x)

Indirect Inconsistency

Rule 3: TEACHING_ASSISTANT(x)$\rightarrow$TENURED(x)

Rule 4: TEACHING_ASSISTANT(x)$\rightarrow$STUDENT(x)

Rule 5: STUDENT(x)$\rightarrow \neg$TENURED(x)

Numeric Completeness

In checking for numeric completeness the logic checker will issue a warning when gaps in a specified numeric range occur. If the rule base contains only rules 6 and 7 (dealing with the number of items in a discounted order),

Rule 6: DIS-ORDER(x) $\wedge \leq$(NUM-ITEMS(x),50)
$\rightarrow$ MIN-DISCOUNT(x)

Rule 7: DIS-ORDER(x) $\wedge \geq$(NUM-ITEMS(x),75)
$\rightarrow$ MAX-DISCOUNT(x)

then the logic checker will propose the rule α to complete the numeric range.

Rule α: DIS-ORDER(x) $\wedge >$(NUM-ITEMS(x),50)
$\wedge <$(Num-Items(x),75) $\rightarrow$??

EXTENDED LOGIC CHECKER

The purpose of the extended logic checker is to check for inconsistencies and conflicts caused by generalization hierarchy, incompatibility, or synonymy. If the application contains rules that can derive contradictory conclusions from the same set of facts and from the properties of generalization hierarchy, incompatibility, or synonymy, then the application is inconsistent.

Inconsistency Under Generalization Hierarchy

Given the metafact that a submarine is a subclass of ship, rules 8 and 9 are inconsistent.

Rule 8: $E \wedge F \rightarrow \neg$send-ship.
Rule 9. $E \wedge F \rightarrow$ send-submarine.

Inconsistency Under Incompatibility

Given the metafact that boy and girl are incompatible, rules 10 and 11 are inconsistent.

Rule 10: $A \wedge B \rightarrow$ is-boy
Rule 11: $A \wedge B \wedge C \rightarrow$ is-girl.

Inconsistency Under Synonymy

Given the metafact that submarine and sub are synonymous, rules 12 and 13 are inconsistent.

Rule 12: $E \wedge F \rightarrow$ send-submarine.
Rule 13: $E \wedge F \rightarrow \neg$send-sub.

In each of the above examples, the conditions in each of the rule pairs are exactly the same or one condition is a proper subset of another. However, the inconsistency can occur in a more general case. The examples below are in **conflict** if **all** of the conditions A, B, C, and D are met in the knowledge base **at the same time**.

Conflict Under Generalization Hierarchy

Given the metafact that submarine is a subclass of ship, rules 14 and 15 are in conflict.

Rule 14: $A \wedge B \rightarrow \neg$send-ship.
Rule 15: $C \wedge D \rightarrow$ send-submarine.

Conflict Under Incompatibility

Given the metafact that boy and girl are incompatible, rules 16 and 17 are in conflict.

Rule 16: $A \wedge B \rightarrow$ is-boy
Rule 17 $C \wedge D \rightarrow$ is-girl

Conflict Under Synonymy

Given the metafact that submarine and sub are synonymous rules, 18 and 19 are in conflict.

Rule 18: $A \wedge B \rightarrow \neg$send-sub
Rule 19: $C \wedge D \rightarrow$ send-submarine

If the conditions A, B, C, and D can be satisfied by future facts in the application (recognized from assertions in the RHS of rules), the logic checker warns of *potential conflict*.

STRUCTURE CHECKER

The kernel module of the structure checker is an analyzer which takes in the facts and rules of a knowledge-based application and builds a rule graph. An arc in the rule graph denotes a match between a relation in the LHS of a rule and a relation in the RHS of a rule. Other structure checker modules analyze the rule graph to locate all the structural errors in the knowledge base, such as nonreachability, redundant rules, irrelevant rule clauses, and rule cycles.

Reachability

The purpose of reachability checking is to determine if the rule graph has any missing subgraphs or disconnected subgraphs.

In checking for reachability, EVA looks for dead-end nodes and unreachable nodes. A node here means an atomic formula in a rule. A "dead end" occurs when a node in the LHS of a rule does not match any fact, goal, schema, or RHS rule node in the knowledge base. An unreachable node is a fact, goal, schema, or RHS rule node that cannot be matched by any LHS rule node in the knowledge base.

Redundancy

A rule is redundant if it is duplicated or subsumed by another rule. Two rules duplicate one another if they have the same LHS and RHS clauses, possibly in different order or with different variable names. Subsumption occurs when two rules have duplicate RHSs and the LHS of one is a subset of the LHS of the other, or when two rules have duplicate LHSs and the RHS of one is a subset of the RHS of the other, or a mixture of both.

Duplication:
Rule 20: MALE(x)$\wedge$ PARENT(x) $\rightarrow$ FATHER(x)
Rule 21: PARENT(y)$\wedge$ MALE(y) $\rightarrow$ FATHER(y)

Subsumption:
Rule 22: TENURED(x)$\wedge \neg$STAFF(x)
$\rightarrow$ UNIVERSITY_MEMBER(x)
Rule 23: TENURED(x) $\rightarrow$ UNIVERSITY_MEMBER(x)

Relevance

Two rules contain irrelevant or superfluous clauses if they are duplicates except that in one LHS a clause p is affirmed, in the other denied.

The structure checker reports that the first clause in rules 24 and 25 is irrelevant.

Rule 24: TEACHING_ASSISTANT(x)∧STAFF(x)
→UNIVERSITY_MEMBER(x)

Rule 25: ¬TEACHING_ASSISTANT(x)∧STAFF(x)
→UNIVERSITY_MEMBER(x)

Cycles

EVA checks for direct and indirect cycles in the rule base. A direct cycle exists when the same clause occurs on both the LHS and RHS of a rule. This may cause the rule to fire repeatedly -- that is, result in an infinite loop. In the case of an indirect cycle the infinite loop involves two or more rules.

Direct Cycle:
Rule 26: ANCESTOR(y,x)∧PARENT(x,z)
→ANCESTOR(y,z)

Indirect Cycle:
Rule 27: HUMAN(x)→PERSON(x)
Rule 28: PERSON(x)→HUMAN(x)

EXTENDED STRUCTURE CHECKER

The extended structure checker checks for duplication, subsumption, relevance, and cycles caused by generalization hierarchy or synonymy.

Duplication

Given the metafact that sub and submarine are synonymous, rules 29 and 30 are duplicates of one another.

Rule 29: SUB(x)→DIVE(x)
Rule 30: SUBMARINE(x)→DIVE(x)

Subsumption

Given the metafact that submarine is a subclass of ship, rule 31 is subsumed by rule 32; i.e., whenever 31 is applied, 32 will also be applied.

Rule 31: SUBMARINE(x)→LAUNCH(x)
Rule 32: SHIP(x)→LAUNCH(x)

Relevance

Given the metafact that sub and submarine are synonymous, the first clause in rules 33 and 34 is irrelevant.

Rule 33: SUBMARINE(x)∧F(x)→E(x)
Rule 34: ¬SUB(x)∧F(x)→E(x)

Indirect Cycle

Given the metafact that sub and submarine are synonymous, rules 35 and 36 create an indirect cycle.

Rule 35: SUBMARINE(x)→G(x)
Rule 36: G(x)→SUB(x)

SEMANTICS CHECKER

The semantics checker has two major functions: checking the application written in the object shell for violations of the semantic constraints, and checking the semantic constraints themselves for internal consistency and agreement with other metaconstraints represented in the metashell.

EVA's metarelations take object, class, slot and relation names in the object shell as arguments; they permit the knowledge base designer to specify semantic constraints (conditions) that must be met by the underlying application. Facts violating the semantic fact constraints are either incorrect facts or show that the metaconstraint itself is not correct, while rules violating the semantic rule constraints are either incorrect rules or show that the *metaconstraint* is not correct.

In particular, the semantics checker checks facts in the application for violation of range constraints, minimum/maximum cardinality constraints, legal value constraints, value compatibility constraints, and for other semantic constraints such as subrelations, inverses, and data types.

The semantic constraints that have been implemented in an ART-based prototype are explained below. More semantic constraints -- circular ranges (Sunday, Monday,...; 00:00-24:00 hours; Jan,...Dec; and so on), and nonnumeric ranges (cold, warm, hot; Private, Corporal, Sergeant,...; and so on), as well as relations between relations/attributes/sets such as inverse, synonymous, antonymous (male/female, i.e., not male implies female and vice versa), contrary (young/old, i.e. not young does **not** imply old), compatible, incompatible, and others -- will be implemented in 1987.

lower_upper(slot, class, lower, upper)

This metarelation defines the legal range of numerical values: the values for the <slot> of the <class> must be between <lower> and <upper>. EVA discovers and flags values that exceed these bounds.

EVA not only checks facts against semantic constraints, but also checks that the semantic constraints themselves are consistent. Since a CHILD is a PERSON, the age range of CHILD must fall within the age range of PERSON. Thus the following semantic constraints would be inconsistent:

```
  is_a(CHILD, PERSON)
  lower_upper(AGE, PERSON, 1, 110)
  lower_upper(AGE, CHILD, 0, 12)
```

legal_value(slot, class, values)

This metarelation defines the legal values of a slot: the values for the <slot> of the <class> must be listed in <values>.

For example, the following semantic constraint

```
  legal_value(GENDER, STUDENT,
        [male,female,hermaphrodite])
```

states that students' genders must be male, female, or hermaphrodite. EVA flags any STUDENT record where GENDER has a nonlegal value.

relation(rel,domain-1,...,domain-n)

This metarelation defines both the number of legal arguments of a relation <rel> and the legal data type of each argument: <domain-i> is either a class of objects or a set of values for i=1,...,n.

This kind of semantic constraint is used to permit EVA to enforce strong data-type checking for relations.

For example, given the two facts

```
  PERSON(charlie)
and
  DOG(snoopy)
and the metafact
  relation(MURDERER_OF,PERSON,PERSON)
EVA flags as erroneous the fact
  MURDERER_OF(charlie,snoopy).
```

compatible_arg_types(rel,argument_value_list)

This metarelation defines the legal value combinations for individual argument types. Thus the metafact

```
  compatible_arg_types(ENROLL,
     ([arg1 (FRESHMAN)
          arg2 (Math101,English101,...)]
      [arg1 (SOPHOMORE)
          arg2 (Math201,Art201,...)]


     ))
```

states that a FRESHMAN can only ENROLL in Math101, English101, ... that a SOPHOMORE can only ENROLL in Math201, Art201, ... and so on.

EVA also checks the defined inverse R' of a relation R for agreement with argument data type specifications and legality of argument value combinations using the metarules stated for R.

min_max_rel(relation, domain, min, max)

This metarelation specifies the minimum and maximum number of tuples (records) of a relation: the number of records of <relation> with object in <domain> must be between <min> and <max>.

For example, the following semantic constraint

```
  min_max_rel(ENROLL, STUDENT, 0, 5000)
```

means that up to 5000 student enrollments are allowed in the data base at any one time. The enrollments are represented by records or tuples of the relation ENROLL.

min_max_role(relation, domain, min, max)

This metarelation defines that each object in <domain> must have at least <min> and at most <max> objects for the relation <relation>.

Thus the semantic constraint

```
  min_max_role(ENROLL, SOPHOMORE, 3, 4)
```

states that each SOPHOMORE must ENROLL in at least 3 and at most 4 courses. EVA discovers any sophomore who enrolls in fewer than three or more than four courses.

incompatible(slot, class, values)

This metarelation defines **illegal** value combinations: the multivalue <slot> of <class> must not have the combinations listed in <values>.

For example, the semantic constraint

```
  incompatible(TITLE, UNIVERSITY_MEMBER
       [(Tenured,Untenured),
       (Tenured,Student),
       (Tenured,Staff)])
```

means that no UNIVERSITY_MEMBER can have both the TITLE Tenured and Untenured, Tenured and Student, or Tenured and Staff.

compatible(slot, class, values)

This metarelation defines **legal** value combinations: the multivalue <slot> of <class> may contain the combinations listed in <values>.

For example, the semantic constraint

compatible(TITLE, UNIVERSITY_MEMBER
[Untenured, Student, Staff])

means that a UNIVERSITY_MEMBER may have up to three TITLES from the list Untenured, Student, or Staff. [This metarelation complements the previous one and is to be used when the number of illegal combinations would be too tedious to list.]

subrelation(relation1, relation2)

This metarelation defines that <relation1> is a subrelation of <relation2>. EVA checks that the number of arguments for <relation1> and <relation2> are the same, and that the data types of the arguments of <relation1> are subclasses of the corresponding arguments of <relation2>.

For example, EVA determines that the semantic constraints

relation(KILLER_OF,
ANIMATE_OBJ, ANIMATE_OBJ)
relation(MURDERER_OF,
PERSON, THING)
subrelation(MURDERER_OF, KILLER_OF)

are inconsistent since the second argument (THING) of MURDERER_OF is not a subset of the second argument (ANIMATE_OBJ) of KILLER_OF.
EVA also checks that the inverse of <relation1> is a subrelation of the inverse of <relation2>. EVA also creates the missing inverse of a subrelation, if one does not exist.

FUTURE EVA FUNCTIONALITY

The functional components described below are part of our current research and will be implemented in future EVA prototypes.

Omission Checker

Very often an application written in the object shell is incomplete. This means there are omissions in the application knowledge base. The omissions may be related to the structural rule graph, e.g., omission of rules for missing cases or omission of terminating rules. These are similar to missing some case or terminating statements in a conventional programming language.

The other facet of the omission checker is related to induction or machine learning. Through the recognition of structural information contained in facts and rules in the application and semantic information contained in the metaknowledge base, the omission checker may identify some relations that are not defined by the user, or learn rules to associate certain conditions with certain classes of objects.

For example, if the following information is known

Class of objects: person, man, woman;
Generalization hierarchy:
man isa person, woman isa person;
Metafact:
person is union of man and woman;
Relation schema:
PARENT-OF(person,person);
Relation schema:
FATHER-OF(man,person);
Metafact:
FATHER-OF is subrelation of PARENT-OF;

then the omission checker can prompt the user whether there should be a relation that relates woman and person, in analogy to FATHER_OF which relates man and person, (namely, MOTHER-OF).

Rule Proposer

There are two ways to use the rule proposer. The user can use it to simplify a collection of rules through induction. That is, some conditions appearing in a collection of rules may be abstracted by induction into a more general condition, and thus the collection of rules can be simplified into a single rule. For example, in rules 37, 38, and 39,

Rule 37: $\text{LOSS\_CANOPY}(x) \wedge \text{ALTITUDE}(x) > 20000$
$\rightarrow \text{REDUCE-ALTITUDE}(x)$.
Rule 38: $\text{LOSS\_OXYGEN}(x) \wedge \text{ALTITUDE}(x) > 20000$
$\rightarrow \text{REDUCE-ALTITUDE}(x)$.
Rule 39: $\text{LOSS\_ENGINE}(x) \wedge \text{ALTITUDE}(x) > 20000$
$\rightarrow \text{REDUCE-ALTITUDE}(x)$.

the relations dealing with the "loss" of a system part are abstracted into a more general relation, e.g., CRIPPLE,

Rule α: $\text{LOSS\_CANOPY}(x) \rightarrow \text{CRIPPLE}(x)$
Rule β: $\text{LOSS\_OXYGEN}(x) \rightarrow \text{CRIPPLE}(x)$
Rule γ: $\text{LOSS\_ENGINE}(x) \rightarrow \text{CRIPPLE}(x)$

and the three rules (37, 38, 39) are reduced to one general rule.

Rule ρ: $\text{CRIPPLE}(x) \wedge \text{ALTITUDE}(x) > 20000$
$\rightarrow \text{REDUCE-ALTITUDE}(x)$.

On the other hand, the rule proposer can be used in an intelligent debugger to adjust rules to fit test cases. When the application programmers run the application on the test cases, they may find errors in the application caused by incorrect rules. Conditions in the LHS of a rule usually define a class. In the case of an incorrect rule, the rule proposer may apply the restriction or generalization operation on the rule to define a more specific or more general class, respectively. The modified rule may allow the application to pass the test cases successfully.

Behavior Verifier

A system may be decomposed into many subsystems. A subsystem may be represented by a collection of facts and rules in the object shell. However, the subsystem must have external input/output interfaces to communicate with the outside world. For example, in the space shuttle flight software system, the navigation controller is a subsystem that sits in a control loop, collects and analyzes data, and then sends control signals to the vehicle manipulator.

The behavior of the subsystem is a description of relationships among the external input/output interfaces and internal states of the subsystem. The subsystems are connected together to make the system. The verifier is to prove that the intended behavior of the system can be derived from the behaviors of the subsystems and the connection descriptions.

Control Checker

All existing object shells provide some control constructs to sequence the firing of rules in the application. For example, in ART, facts acting as permitting conditions and *salience* (priority instructions) can be used to control the rule firing sequence.

In EVA, if the system designers want to impose ordering constraints on some actions, they can specify them in the metashell, and the control checker will verify if the order specified in the application rules corresponds to or violates the ordering constraints specified in EVA's metacomponents For example, in an office system, an ordering constraint is that a paper must be "cleared" before it is "published."

CONCLUSION

The first prototype of EVA has been implemented in ART and LISP. ART-based systems being developed at Lockheed such as the Software Project Management system[7] have been used as test cases. Other Lockheed companies are using our prototype for their knowledge-based applications.

It is evident that EVA provides a powerful means for representing knowledge about an application domain and for verifying that the knowledge is correct and complete. EVA increases the reliability of knowledge-based systems, speeds up their development, and assists in their continuing modification. The necessity for such validation tools will continue to grow as future knowledge-based systems play a more critical role in business, industry, government, and the sciences.

References

1. R.A. Stachowitz, J.B. Combs, "Validation of Expert Systems", *Proceedings of the Twentieth Hawaii International Conference on Systems Sciences*, 1987, pp 686-695.

2. E. Miller, W. Howden, *Tutorial on Software Testing and Validation Techniques*, IEEE, New York, NY, 1981.

3. T.A. Nguyen, W.A. Perkins, T.J. Laffey, D. Pecora, "An Expert Systems Knowledge Base for Consistency and Completeness" *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp 375-378.

4. M. Suwa, A.C. Scott, E.H. Shortliffe, "An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System", *The AI Magazine*, Vol. 1982, pp. 16-21

5. W.W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques", *Proceedings*, WESCON, August 1970.

6. B.W. Boehm, "A Spiral Model of Software Development and Enhancement", *ACM Software Engineering Notes*, March 1986.

7. K.D. Bimson, L.B. Burris, "Knowledge Representation in Software Project Management: Theory and Practice", *Proceedings of the Forum on Artificial Intelligence in Management*, May 1986.

AFOSR · TR · 89 - 1600

AIAA-87-1686
New Concepts in Tele-Autonomous Systems

L. Conway, R. Volz and M. Walker,
University of Michigan,
Ann Arbor, MI

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program
March 9-11, 1987/Arlington, VA

NEW CONCEPTS IN TELE-AUTONOMOUS SYSTEMS*

Lynn Conway, Richard Volz and Michael Walker

Robotics Research Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109

ABSTRACT

We have taken up the challenge of integrating telemanipulation technology and autonomous system technology. We are seeking methods for integration at a fundamental rather than an ad-hoc level. We believe that success in this effort can open up new space and defense applications now beyond reach of either technology alone.

In our presentation at this symposium, we introduce a series of concepts for "tele-autonomous" systems. The concepts involve new system architectures and associated new system interface controls including "time clutches", "position clutches" and "time brakes".

Taken together, the concepts enable effective, efficient intermingling of real-time cognition and manipulation tasks performed by either humans or machines. The concepts also yield simple mechanisms and protocols for handoffs of such tasks between multiple agents.

In this presentation we focus primarily on the tutorial introduction of the basic tele-automation concepts. We then briefly describe our environment for exploring this new technology and the results of our initial experiments. Further details concerning tele-autonomous syste . architecture and our initial experimental results can be found in an attached reference [CON87].

* This presentation is based on recent work described in a paper [CON87] to be published in the Proceedings of the IEEE International Conference on Robotics and Automation, March 30, 1987. A preprint of that paper is included with this AIAA/NASA/USAF Symposium preprint. Our presentation is further supplemented with a University of Michigan Robotics Research Laboratory video-report [CON87a].

INTRODUCTION

We are seeking simple, generic methods for intermingling and integrating telemanipulation and autonomous systems technology. Now, you might ask, why would we want to do that?

First, we wish to provide more effective systems for autonomous environmental manipulation. Consider an AI cognition system embedded within an overall perception-cognition-action system. Many tasks of interest will involve perception- cognition-action computing delays on the order of fractions of a second or seconds. How can we deal with such delays, when the basic behavioral acts to be done to complete a manipulation task themselves require only on the order of fractions of a second or seconds?

Presently we require substantial environmental knowledge and then piece together preprogrammed forms of interactions to cope with such delays. When that isn't possible, we fall back on a rather halting, stumbling form of perceive-think-act cycling, where the perception to action delays are contained in each basic behavioral act. Could we get around this somehow? After all, animals often perform manipulations with the aid of visualizations out just in front of their real time actions. Could we mechanize something like that?

The second challenge is to provide protocols for the interaction between multiple autonomous manipulation agents. Consider an ALV driving down a remote road. It suddenly encounters uncertain footing, and doesn't have sufficient exploratory behaviors and learning capabilities to get itself out of trouble. We know that AI will not soon be able to handle all the cognitive tasks and especially not all the manipulation tasks to get an ALV out of this kind of trouble. But how can we enable a human to easily "slip into the cockpit" and take over in mid-manuever in such situations? How are they to perceive the local task goals and judge whether they are making progress towards them? And how are they to hand the task back to the machine?

This intriguing example illustrates the need to bridge the gap between direct human control and AI control of manipulation systems, and the need for handoff protocols between autonomous agents, whether human or machine. It is suggestive of the larger space of such examples, many of which are emerging in the machine intelligence research programs now underway under DARPA's Strategic Computing Initiative [DAR83], and emerging as a result of NASA space station automation activities.

The basic challenge in such examples is how to structure the actions and interactions of intelligent, goal-seeking systems when their activities are at least in part based on their physical manipulations of their environment rather than simply on symbolic communication. The challenge is how to mediate the interactions of cognitive agents that are embedded in perception-cognition-action systems.

Certainly there are a lot of ad-hoc, task-specific systems being built that do interact effectively in very narrow contexts. But such ad-hoc systems, while individually useful, do not readily generalize to provide a basis for others to build upon. Could we find some simple, general ways to think about the problem? Could there be some basic, generic protocols on top of which we could construct more uniform systems? We think the answer to these questions is yes.

BASIC TELE-AUTOMATION CONTROLS

We now present a sequence of interface control concepts that collectively enable efficient control of manipulation tasks and that enable simple protocols for exchange of such tasks between control agents. This paper focusses on tutorial development of the basic functional ideas. For details on architectures to mechanize these concepts, see [CON87].

We begin by looking via video link over the shoulder of a telerobotic manipulator, and controlling the manipulator via a joystick as shown in Figure 1. We are to perform the simple task of touching in sequence each of a series of boxes. This task's difficulty is some function of the ratio of the distance between consecutive boxes and the sizes of the boxes. The difficulty can be varied easily, and we can undertake various trials of performance as a function of system parameters. For example, we could do some simple trials to see if the time to complete the task is a logarithmic function of D/S, as in Fitt's Law [CAR83].

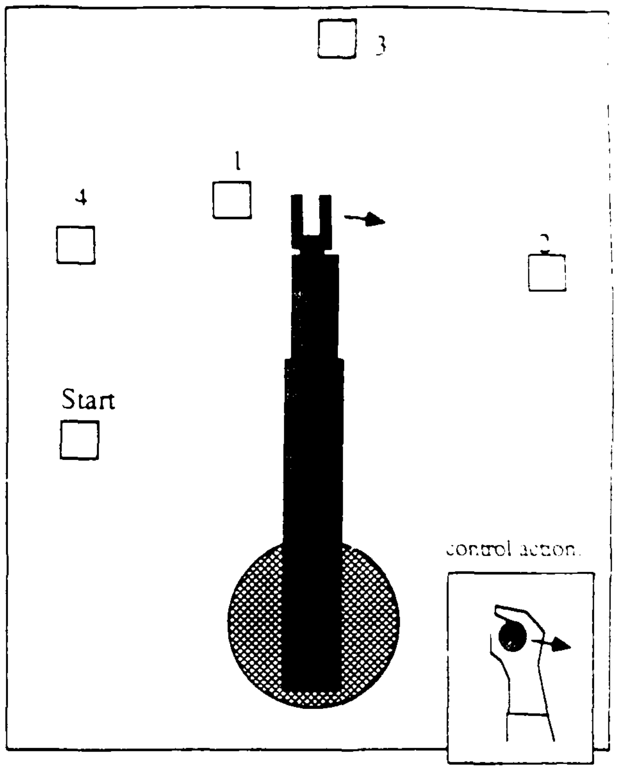


Figure 1. Visualizing a remote manipulation task

Coping with Time Delay

Because of our interest in remote space and defense systems, we visualize trying the same manipulation with a time delay inserted into the return video path. We find that the telerobot's motions then tend to be rather slow and jerky. The operator must move a little and then wait through the time delay to see what happened. As an illustration of this point, observe in the video-report the telerobot performing the manipulation task with no delay and then with a 2.0 second delay. The difficulties introduced by the time delay are quite noticeable. The time to complete the task is greatly extended. (Note that for convenience, we display a model of the telerobot using a Silicon Graphics IRIS workstation. The model is driven by the actual joint angles of the telerobot, and is thus equivalent to observing video of the real telerobot for our purposes here).

To overcome the time delay problem, Noyes and Sheridan [NOY84] have suggested that the operator control a local simulation of the telerobot, with the control signals then sent in parallel to the simulation and the remote telerobot. The simulation is then displayed superimposed over the return video. In this way the operator can "see" the effects of the control immediately without having to fully wait for the return signal from the telerobot. As a result, task time is reduced to nearly that of the no-delay case. (Noyes's and Sheridan's concept is further sketched in figures 2.1. and 2.2 in [CON87]).

Figure 2 presents a visualization of telerobotic manipulation using a forward simulation to cope with the time delay, as does the next segment in the video-report. The wire frame is the forward simulation that directly responds to operator control, and the solid frame represents the time delayed image of the real telerobot. Much faster and smoother control is achieved, as is evident in the videotape. This is a first step towards evolving machine manipulation visualization, since the visualization could help cope not only with communication delays, but also with computational delays within a self-contained autonomous agent.

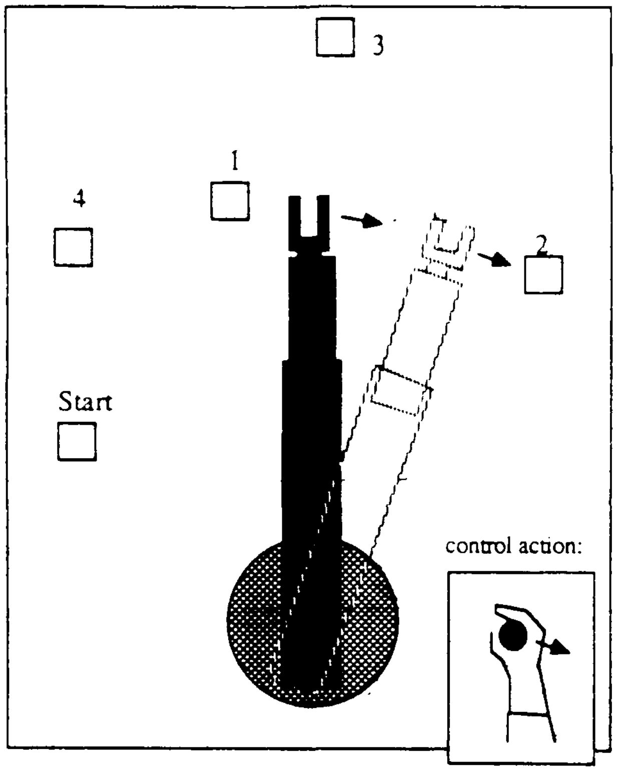


Figure 2. Visualizing manipulation through a time delay using forward simulation.

The Time Clutch

But, this first exploitation of real-time forward simulation is only the beginning. Forward simulation can also be exploited even if we don't have a communications time delay. To do this, we introduce the concept of a "time clutch" to disengage synchrony between operator specification time and telerobot manipulation time during path specification. Our hypothesis is that operators can often think of and generate a path segment more quickly than the telerobot can follow it. Once generated, such a path segment can then be followed more quickly by the robot than would be the case if the robot were time-synchronized to the specification process: with time synchrony disengaged, the robot can steadily proceed at nearly its maximum rate, subject of course to error limits and hard constraints.

An overall diagram of the basic system architecture including the time clutch is contained in figure 3.1 in [CON87]. Figure 3 in this presentation shows a path being generated well out in advance of the actual robot by an operator using forward simulation with time clutch disengaged. The associated video-report also demonstrates the effects of disengaging the time clutch; if you put a stop watch on the action, you will measure a significant speed up of the real telerobot's motion from that obtained using forward simulation alone.

This step in the evolution of machine manipulation visualization enables the cognitive agent to "look and think ahead" of the manipulation under control, with the look-ahead time being elastic, and not just a fixed internal or external system time delay. The implementation of this new capability requires only a simple mutation of the forward simulation previously used for coping with a time delay.

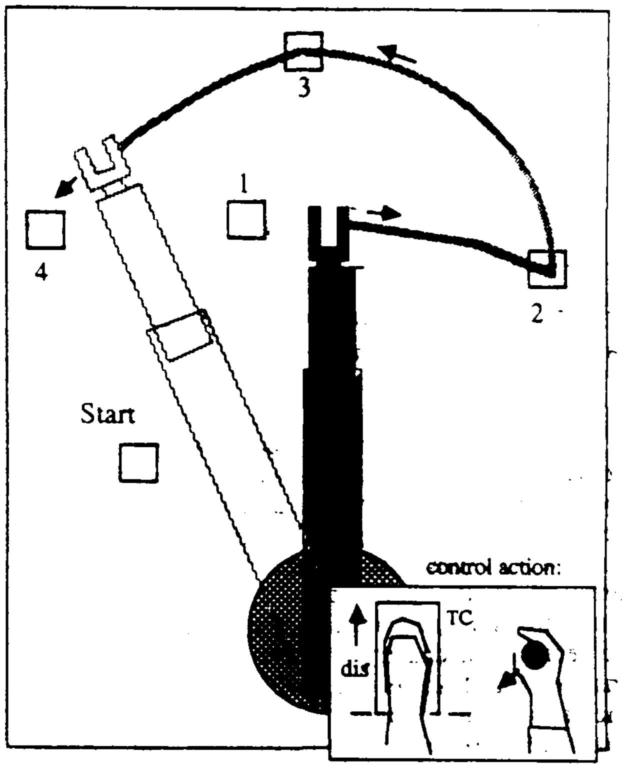


Figure 3. Rapid manipulation path generation using forward simulation with time clutch.

The Position Clutch

We next introduce the concept of a "position clutch" which enables a disengagement of position synchrony between simulator and manipulator path (see figure 3.1 in [CON87] for system diagram). We hypothesize that faster, shorter, cleaner paths can be generated on difficult tasks using this control. This idea is illustrated in Figure 4, which shows the use of the position clutch to disengage from path generation during a close approach to a difficult manipulation (in this case, touching a small object).

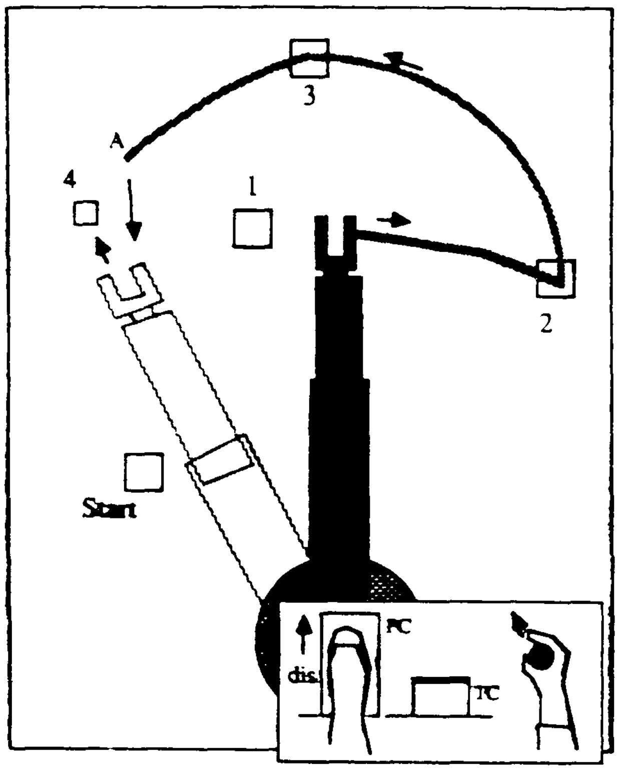


Figure 4. Using the position clutch to cope with a more difficult manipulation.

Suppose, for example, that the operator had arrived (in the simulation) at point A ahead of time by using the time clutch. The position clutch can then be disengaged, stopping the output from the operator control from going to the real telerobot - - it will only go to the simulation. When the forward simulator is in good position, the position clutch will be reengaged, causing a short, smooth path to be inserted that links to the earlier path. This avoids inclusion of jittery prepositioning movements in the final path to be followed. Further, the time spent by the operator in achieving the proper position will not be incurred by the real telerobot since these motions were "clipped" out of the path sent to the telerobot.

The operator has thus used up some of the time saved through use of the time clutch, with the result that the overall task time of the telerobot is reduce still further. The next segment in the video-report illustrates this point. This level of manipulation visualization corresponds to quick "visualizations and visualized trials of multiple alternatives" prior to commitment to action, and its implementation requires only another simple mutation of the basic forward simulation capability.

The Time Brake

To handle contingencies and errors we introduce the concept of a time brake. This control can be used to deal with situations such as something falling over a previously generated path, as illustrated by the "X" in Figure 5. In Figure 5 we see the time brake being applied and the forward-simulated manipulator backing down the path (in a race to get on the other side of the obstacle before the real system gets there). See figure 3.1 and the text in [CON87] for the associated system architectural concepts. The next segment of the video-report demonstrates the application of the time brake.

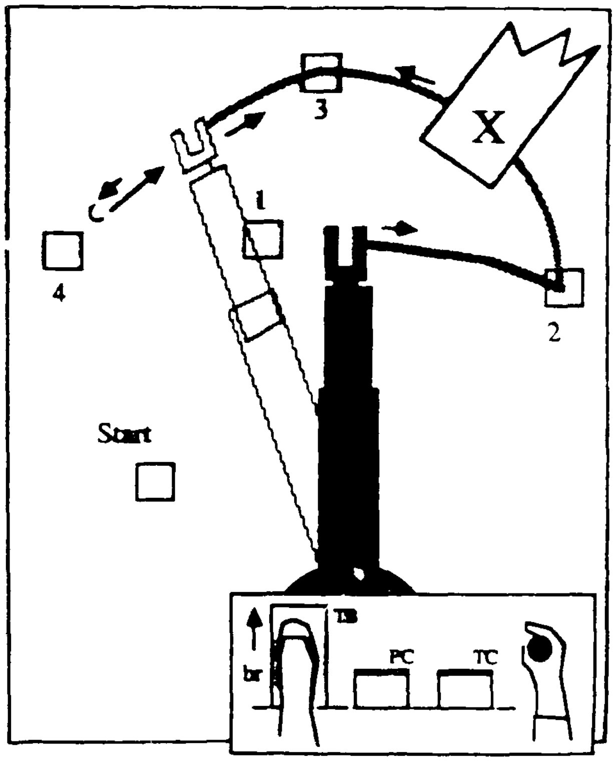


Figure 5. Using time brake to handle a contingency.

This aspect of visualization corresponds to seeing something about to happen that will interrupt an action previously visualized but not yet underway. If it had gotten underway, or is allowed to get underway, the system will have to deal with it through local reflex action or crash. But if visualized in time, the cognitive agent can withdraw the action using the time brake.

TASK HANDOFFS AND RENDEZVOUS

These basic tele-autonomous system interface controls enable us to greatly improve telemanipulation performances, as we'll see in the discussion of our initial experimental results. But the controls do more than that. They also provide the basis for a simple, elegant protocol for hand-offs and rendezvous of tasks between different control agents.

Imagine two operators, one in control of the telerobot and the other about to take over in relief of the first. Each operator would be in control of a simulation of the the telerobot, but only the controls signals of the first would be sent to the real telerobot. The relief operator would, with position clutch disengaged, guide his/her simulation as close to the first operator's as possible (or as close as required, as a function of the interpolation and smoothing methods to be used in the rendezvous). The first operator then disengages their position clutch, leaving the path "hanging". Figure 6 shows this moment in the interaction.

The second operator then engages their position clutch, rendezvousing with the path and taking control of future path generation. When the actual manipulator passes over this path segment, it will do so smoothly and will not notice that a change of control agent has occurred in mid-manuever. We can again find interesting biological analogies to this visualization situation. For example, consider the interactions among basketball players as they previsualize fast-paced multiplayer interactions.

We believe that this simple protocol can be built upon to mechanize quite a wide range of manipulation interactions between autonomous agents. See for example the discussions in [CON87] concerning the various scenarios of "student pilot and instructor pilot", where we consider handoffs of manipulation and cognition tasks among humans, between humans and machines, and among machines.

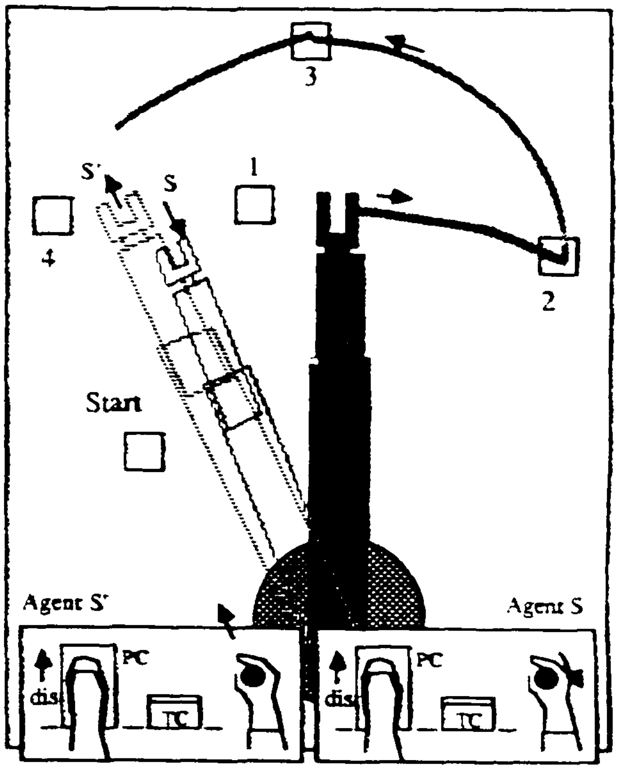


Figure 6. Using time and position clutches to handoff task to another forward simulation agent.

INITIAL EXPERIMENTS AND RESULTS

The manipulation tasks in our initial experimental trials consisted of having the telerobot touch a series of boxes of size S, each separated by a distance D. By varying D and S we produced tasks over a range of difffficulties. Our goals were to test our hypotheses that the time and position clutches can improve the overall manipulation performance, and also to study the detailed functional relationship of task time vs task difficulty over a range of system parameter values. Analyses of such relationships may lead to principles for the design of future tele-autonomous manipulator systems.

We can take a closer look at our experimental setup in the video-report. You'll notice the solid robot image on the Silicon Graphics IRIS display corresponds to the real robot. To simulate the effects of communication delays, we insert a time delay in transmissions of information between the user control and the real telerobot. The user controls the telerobot through a joystick. The time and position clutches and the brake are foot pedals on the floor. While either of the clutches is depressed, the corresponding synchrony is disengaged, and while the brake is depressed, the simulation backs up in time toward the real telerobot's position.

Each square block to be touched is placed a fixed distance D from its predecessor, but at a randomly determined angle. The tests were performed across a a number of subjects, each performing each test a number of times with different random placements of the blocks. Results were first obtained with and without time delays for direct tele-manipulation (with no forward simulation), providing baselines for the remaining trials of the different modes of operation over a range of system parameters. The principle system parameters varied during these trials were:

(i) Difficulty ratio D/S,
(ii) Manipulation size scale D,
(iii) Communication delay time,
(iv) Mode of operation (with/wo clutches),
(v) Robot joint angular velocity limits.

The resulting data on operator task specification times and robotic manipulation times are summarized graphically in figures 5.2 and 5.3 in [CON87]. In those figures we see that the times for manipulation in presence of delays are substantially reduced by forward simulation and then again substantially reduced by use of the new tele-automation controls. (Substantially means a time improvement of at least a factor of two). For some parameter values the use of the time clutch enabled operators to move out far ahead of the following telerobot. The use of the position clutch enabled operators to produce shorter manipulation path lengths on complex tasks. Demonstrations of these trials and also of handoffs using the position clutch are shown in the video-report. Refer to [CON87] for a detailed presentation and analysis of our experimental results.

RESEARCH PLANS AND ISSUES

We are augmenting our experimental environment to enable trials using additional forms of telerobotic manipulators and manipulation interface controls. We are further analyzing the sensory-cognitive-motor dynamics of the human-machine combination in efforts to generate additional testable hypotheses regarding factors affecting performance. We are also planning trials of simple autonomous activity, with the manipulations and handoffs being done under the control of AI planning programs.

This early tele-automation work suggests many opportunities for new interdisciplinary interactions among those interested in human-computer interaction, robotics, and artificial intelligence. There are also challenges and opportunities concerning provision of appropriate experimental environments for such work in the larger research community. The provision of shared access to remote telemanipulation facilities might enable more researchers to collaborate on the evolution of such new technologies [CON87].

SUMMARY

We have begun to explore the integration of telemanipulation and autonomous system technology. We have created new concepts that enable substantial telemanipulation task performance improvements. These improvements are applicable to manipulations controlled by humans or by machines. Our initial hypotheses concerning performance improvements are well supported by our initial experiments. We have also demonstrated that the new concepts also enable simple, elegant protocols for handoffs of manipulation tasks between autonomous agents. We believe that a number of useful new space and defense applications can be based upon these concepts.

REFERENCES

[CAR83] Card, S., Moran, T. and Newell, A., *The Psychology of Human-Computer Interaction*, Lawrence Elbaum Assoc., Hillsdale, NJ, 1983.

[CON87] Conway, L, Volz, R. and Walker, M., "Tele-Autonomous Systems: Methods and Architectures for Intermingling Autonomous and Telerobotic Technology", *Proceedings of the IEEE International Conference on Robotics and Automation*, March 30, 1987.

[CON87a] Conway, L., Volz, R. and Walker, M., "New Concepts in Tele-Autonomous Systems", Robotics Research Laboratory Video-Report, University of Michigan, February, 1987.

[DAR83] DARPA, *Strategic Computing: New Generation Computing Technology - A Strategic Plan for its Development and Application to Critical Problems in Defense*, Defense Advanced Research Projects Agency, Arlington VA, October 28, 1983.

[NOY84] Noyes, M. and Sheridan, T., "A Novel Predictor for Telemanipulation through a Time-Delay", *Proc. of the Annual Conference on Manual Control*, NASA Ames Research Center, Moffett Field, CA , 1984.

TELE-AUTONOMOUS SYSTEMS: METHODS AND ARCHITECTURES FOR INTERMINGLING AUTONOMOUS AND TELEROBOTIC TECHNOLOGY

Lynn Conway, Richard Volz and Michael Walker

Robotics Research Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, Michigan 48109

ABSTRACT

As a result of recent advances in artificial intelligence, human cognitive modelling, autonomous systems and telerobotics, there is an opportunity to broaden our concepts of technology for projecting action at a distance. We draw on these advances to develop a conceptual and architectural framework that enables efficient projection in time and space of intermingled manipulation and cognition tasks.

Where AI-based autonomous systems have previously been concerned with human supervisory intervention primarily at a cognitive level, we add methods for rendezvous, capture and rehandoff of embedded manipulation tasks. Where telerobotics has been concerned with the projection of sensory-motor manipulation, we add the projection of cognitive processing. Thus extended, the two technologies mirror one another and merge into one of "tele-autonomous systems".

We introduce notions of how the sensory, cognitive and motor functions of tele-autonomous systems can be factored and transferred back and forth between human and machine. We illustrate how the times to complete tele-autonomous tasks can be reduced through time and space constraint relaxations effected through simple controls: We employ the concepts of forward simulation and predictor display, augmented by "time and position clutches", "time ratio controls" and "time brakes", to control the resulting manipulation paths and event transitions. We sketch some generic architectural and human interface implications of these methods. Finally, we describe our environment for exploring these methods and the results of some recent experiments.

1 INTRODUCTION

In this paper we draw on recent research advances in autonomous systems and telerobotics, and develop a framework for an integrated technology that enables efficient projection in time and space of intermingled manipulative and cognitive tasks. The technology builds a bridge between telerobotics and intelligent autonomous systems by providing methods for controlling real-time transitions between human and machine control of remote events.

In this introductory section we reflect on the paradigms of the telerobotics and autonomous systems research communities. We illustrate gaps in the two paradigms, and opportunities for technology integration, by describing practical tasks humans can do that would be hard to implement with either technology alone.

In the later sections of the paper we introduce functional concepts and terminology for "tele-autonomous" or "tele-automation" technology. We present architectural and control methods for implementing these concepts, discuss our experimental environment for exploring these concepts, and finally present the results of some recent experiments.

1.1. Paradigms of Telerobotics

Up till now the concerns of the telerobotics community have been primarily those of the roboticist and control theorist, dealing with sensing the physical environment, measuring positions, forces, and accelerations, and responding with movements and forces to directly manipulate the physical environment. The human provides the cognitive power of the system, with the human's sensory-motor processing intermediated and projected at a distance by the machine.

The coin of the telerobotics realm is manipulation. Support for telerobotics has come primarily from DOE for projection of manipulation capability into hazardous environments. Support has also come from NASA for projection of manipulation into the space environment, and from DoD for undersea applications. When telerobocicists discuss the projection of "autonomous intelligence" to remote mechanisms, the projected capabilities are usually envisioned as programs that can be invoked to independently carry out physical manipulation tasks while the human remains in contact and control at a supervisory level [SHE86].

A common goal of telerobotics research is the production of as realistic a sense of remote telepresence and telecontrol for the human operator as possible, given physical constraints such as communication delay times [NOY84, SHE86]. The goal is to enable operators to do as nearly as well at manipulation tasks as they could do if physically present at the remote location.

1.2 Paradigms of Autonomous Systems

Within the past few years, the U. S. Department of Defense has been supporting a rapidly growing autonomous systems research community [DAR83, DAV85]. This community's concerns are those of computer scientists and artificial intelligence researchers working to produce self contained, mobile platforms, such as the Autonomous Land Vehicle (ALV) [MAR86] and various autonomous undersea vehicles, that can maneuver around and employ machine cognition to seek high-level goals in their environments. The focus is on mechanization of sufficient cognitive power to achieve interesting goals, such as complex route planning and replanning to effect reconnaisance or force projection missions,

Paper to be presented at the 1987 IEEE International Conference on Robotics and Automation.

and on providing sufficient perception and maneuvering capability to do things like follow roads, avoid obstacles, and find things in the environment [DAR83, MAR86].

Given present limitations and computational complexity of self-contained machine perception (such as machine vision), the sensory-motor aspects of autonomous systems technology are currently rather crude when compared to telerobotics, which can exploit human perception. The community thus tends to focus on widening the exploitation of machine cognition on tasks that are feasible given the envelope of available perception technology, conducting a parallel effort on incremental enhancement of perception technology performance [MAR86]. The coin of this realm is cognition, and cognitive interaction with the environment at a symbolic level.

A common goal of autonomous systems research is the mechanization of cognition and the associated task-dependent knowledge systems so that the remote machine is as smart, robust, knowledgeable and persistent as a human might be in attempting to carry out its mission. Since the focus of the work is on autonomy, human supervision or interaction is seldom stressed. When the notion of supervisory control appears in autonomous systems, it usually is concerned with having the human intervene if the system is "not smart enough" to cognitively handle a given situation [MAR86].

1.3. Illustrative task examples.

The following task examples shed light on our problem space, and suggest opportunities and methods for blending telerobotics and autonomous systems. Consider text editing on a workstation. A human operator can often envision and generate the command sequence to achieve a local goal much faster than the workstation can effect the screen manipulation. Thus the human may quickly "type or mouse ahead" (assuming the control stream can be buffered), then shift their cognitive or manipulation attention to the task to be done when the machine catches up [CAR83]. In contrast, in teleoperation systems the operator is often "slaved in real-time" to the local and remote sensory-motor apparatus. Can we imagine a telerobotic analogy to type-ahead?

An extension of "type ahead" occurs when the operator has constructed their own, perhaps intelligent, high-level commands such as "sort this list", or "send a message to X to get the address of Y". The operator may then type ahead at a rather high level, with each command in the sequence performing not just physical manipulations but also elaborate symbolic manipulations of the environment to eventually produce the text. Again, can we imagine a telerobotic analogy?

Or, imagine that you are learning to fly in aircraft that has dual controls. On a given flight your cognition may be just fine, but you suddenly fail to manage a manipulation task, and the instructor takes over. By analogy, an autonomous system might be doing fine in its cognitive tasks, but might need occasional human help in its "lower-level" manipulation tasks. For example, an ALV might run off the road and get stuck, and require a skilled "teledriver" to free it; this is quite a different form of intervention than the "mental" supervisory intervention usually envisioned by ALV'ers.

Intervention into an ongoing autonomous manipulation task may not be easy, since taking over in mid-maneuver may involve smoothly effecting a multidimensional control rendezvous. You can study a simple form of this situation by interacting with the cruise control of your automobile.

The dual-controlled aircraft story yields several scenarios that have interesting autonomous system analogues. The instructor can coach the student on various sensory-motor manipulation tasks, and on various cognitive tasks such as interpreting instrument readings. Visualize the piloting coach as a human supervisor, and the student as a remote autonomous system: The coach can take over either cognition (correcting an instrumentation interpetation) or take over lower-level manipulation (prevent an unwanted stall). There is a matrix of possible division of responsibilities. Sensing, thinking, and acting can be separately assigned and reassigned at any moment to either the supervisor or the system.

But what are the embedded protocols that make such human practices feasible? What shared knowledge is involved? How are the transitions performed? How do both student and coach know who's doing what at any moment? To make the picture even more interesting, consider the fact that the overall system has full duality: the role of coach and student is reversible under some situations. Could such insights have architectural implications for general autonomous systems?

1.4. Merging the augmented paradigms

Can we somehow build a solid bridge between these two technologies so as to merge them? We believe the answer is yes, as discussed in following sections. We also suggest that qualitatively new kinds of functions and new opportunities for performance improvement appear as a result.

2. BASIC TELE-AUTONOMOUS SYSTEM CONCEPTS

Consider either a teleoperator or a supervised autonomous system consisting of (i) a human, and (ii) a machine that is partly local to the human and partly at the remote site of intended projected activity. Among the concerns of architects of such systems are: In specific situations, what is the human best at? Worst at? What is the machine best at? Worst at? How can we shift control between human and machine to exploit these capabilities? Are there generic architectural principles to draw on? What are the constraints on ultimate performance? What traditional constraints can we find ways around? In this section we will explore these questions to develop step-by-step some basic functional concepts for tele-autonomous systems.

2.1. Expanding the functions of telerobotics

Let's first reexamine some of the architect's questions from the "tele-autonomous" point of view. One important constraint on performance in certain key applications is the time delay for communications between the local and remote system. Noyes and Sheridan [NOY84] have innovated and demonstrated a very novel way to cope with such a delay, by using a locally situated forward-in-time telerobot simulator, and a graphical "predictor display" overlay of the forward simulation onto the fixed-delay return video of remote teleoperation (Fig. 2.1). Such forward simulation enables an operator to move the controls and immediately visualize the effect of control action without waiting for the return video. Their experiments show that the time to perform manipulation tasks in the presence of communication delays can be reduced by exploiting such predictor displays (Fig. 2.2).

But instead of just finding ways to better cope with constraints, can we also find ways to relax some constraints? Suppose we had a forward simulator and predictor display, but were not operating through a large time delay. Although we needn't enter commands prior to the observed time of their remote execution, we still might want to do so, and we could

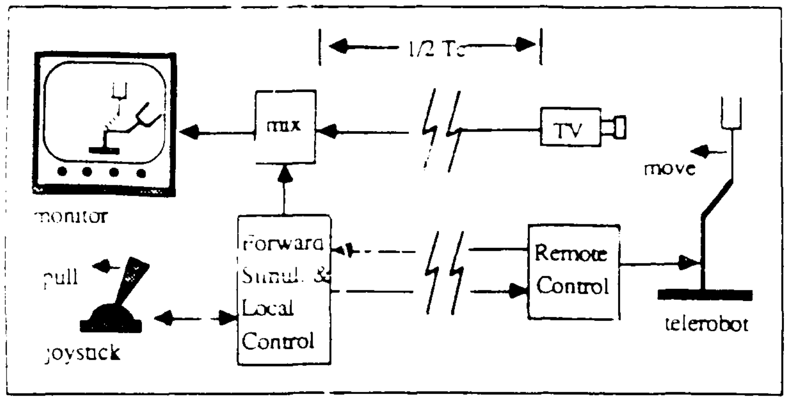


Figure 2.1 Using forward simulation and predictor display to cope with time delay (after Noyes and Sheridan).

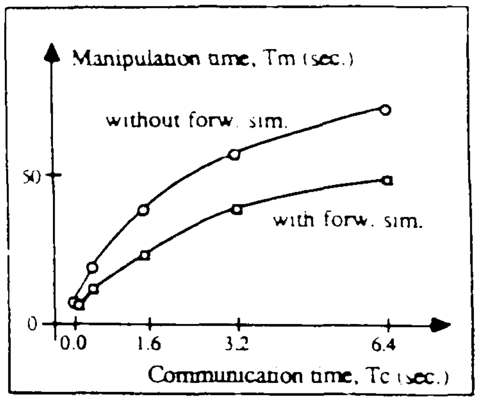


Figure 2.2: Manipulation time as a function of time delay, Tc. [HAS86, NOY84, SHE86]

use a modified version of forward simulation to do so. For example, we might be able to enter commands a lot faster than the telerobot could carry them out, as in "type ahead". Graphical overlay of accelerated forward simulation enables us to do this, and to then manage the cognitive limitations of short term memory when commanding in advance of observed effects. Such simulation can be augmented by including kinematic and/or dynamic forward simulation of portions of the environment. The result is a sort of coordinated "faster than real-time recording, and then real-time playback" form of manipulation control.

There appear to be a number of ways that displays of simulation overlain onto telerobot video can relax telerobotic time-synchrony constraints, yielding possibilities for improvements in manipulation time performance. We also hypothesize that freeing the operator from the "time-slaving" and attention constraining aspects of time-synchronized control may make a qualitative difference in the subjective "feel of the controls" of such systems, making them more like the controlling of one's own limbs. This hypothesis may eventually be made testable by defining new measurements of performance and fatigue in new forms of telerobotic situations.

We can also visualize telerobotic manipulation as analogous to text editing, in that it is a series of sensory-motor limited tasks intermixed with cognitively-limited tasks. Thus in some situations we may be able to project intelligent cognitive functions into the manipulation world (analogous to the "go find the address" command during text editing) while continuing direct editing manipulations. Methods that relax the constraint of command-to-manipulation time-synchrony might enable operators to better intermix such tasks.

2.2. Expanding the functions of autonomous systems

The transfer of cognitive tasks between supervising human and remote machine is already a part of the autonomous systems paradigm, being based on past artificial intelligence work on human machine cooperation in areas like diagnostics, design, advising and coaching [HAY83]. Shifts between machine and human manipulation while the machine retains cognitive control have not usually been considered. However, these can now be seen as just a mirror image, role-reversed version of the augmented telerobotics described above. Any forward simulation, time manipulation and control methods that work there will apply here also. In both cases we must deal with control and human interfacing of rendezvous, capture and rehandoff of manipulation tasks between human and machine.

As we discover control and human interface methods for such manipulation task transfers, perhaps we can also gain insights into how to better structure the methods for cognitive task transfer between human supervisor and autonomous system. Those methods are presently rather ad-hoc, being based on diverse applications experiences in AI. Finally, there is the human interface challenge of presenting "who has control, of what, and at what time and position?" The human may set goals into the autonomous system, and then later be called on to enter tasks to help the system reach either cognitive or manipulation subgoals. Can we use some sort of task lattice or tree, to represent and interface the distributed tasks underway towards goals and subgoals? New human computer interaction knowledge and technology must be developed to support these new possibilities for autonomous systems.

2.3. Integration of tele-autonomous function.

The distinctions between telerobotics and autonomous systems blur when the technologies are each expanded as discussed above. But we don't just get the sum of the two technologies. We get a technology with some new dimensions for enabling action at a distance. This leads us to wonder if we should revise the goal of telerobotics. Could it be possible to project manipulation capabilities to a distance that are better in time performance than those of the unaided human? Considerable research will be required to generate and test hypotheses to determine feasible performance improvements and limits of such an extended tele-manipulation technology. In addition to examining new aspects of robotics and control methods, this research will also enter previously unexplored areas in the psychology of human-computer interfacing

3. TELE-AUTONOMOUS SYSTEM ARCHITECTURE AND CONTROL METHODS.

In this section we describe methods for relaxing the operator to manipulator time-synchrony constraints usually found in telerobotics. The first key idea is the use of a "time-clutch" to enable disengagement of time synchrony during path planning. We extend this idea by adding a "position clutch" that allows forward simulation manipulation and positioning trials without generating path plans. We include a "time-ratio control", to enable variations in the ratio of simulation time to real time. We introduce the concept of a "time brake" to allow the forward simulation to be "braked" back in time to avoid unforeseen contingencies. We then provide scenarios of how these new controls might enable operators of the augmented systems to achieve considerable time improvements in certain manipulation tasks. We also suggest how the augmented architectures enable easy transitions of control of cognition and manipulation tasks between human and machine, thus enabling integrations and mirrorings of telerobotic and intelligent autonomous functions.

3.1. Disengaging time-control synchrony using a Time-Clutch

We build upon the forward simulation and predictor display concept of Noyes and Sheridan as follows. Suppose we are using a telerobotic system as in Fig. 2.1. We augment the system with a control that we call a "Time Clutch". This control enables us to disengage the "direct gearing" or time-rate (but not absolute time) synchrony of simulated time and real-time, and move the forward simulator ahead as fast as skill and judgement will allow. The predictor display presents a forward path as a goal, that is as a sequence of point positions to be followed by the system as fast as is feasible. Note that the path could be generated subject to some settable mean error parameter, for example as a "tube" of given radius [SUH87].

The time clutch enables an operator to disengage from real-time, and manipulate ahead of the displayed video of the real manipulator by working the overlay wire-frame figure of the manipulator on the predictor display. Example situations where this would have benefit would be during slow movements of large space structures and in slow undersea vehicle manipulations. The operator can thus do the telerobotic equivalent of "type-ahead" and then perhaps slow down and carefully position for some tricky maneuver. We hypothesize that in many manipulation task sequences such time saving accumulations and later exploitations will be possible, thus reducing overall manipulation task times and also the fraction of the task time that requires operator involvement.

How can we implement the time clutch control? How is the system to determine the path as a function of time when the clutch is disengaged? The time clutch can be thought of as a simple switch used to make or break the connections within the kinematic/dynamic robot simulation that would normally constrain the rate at which the forward simulator could be slewed around in space. When the clutch is engaged, the position (or rate) joystick control of the simulator is sampled and directly controls the movements of a simulator model which is constrained in its movement rates and accelerations as if it were a real robot. A buffer is inserted between the simulator and the telerobotic manipulator, to hold the stream of sampled position increments as incremental "move to" commands (see Fig. 3.1).

With the time clutch is engaged, the command buffer presents a stream of position points at a fixed sample rate, and the telerobot can simply increment its position accordingly. But when the time clutch is disengaged, the distance between successive path positions may be greater than the telerobot can move in a time sample, and an interpolator is used to generate intermediate points along the path. This interpolator can always be active, with the only difference in function upon disengaging the time clutch being the breaking of simulator contraints on simulator velocities and accelerations. In sophisticated systems where telemanipulator touch sensing and force-sensing during interactions with the environment are reflected back to the manipulator operator, disengagement of the time clutch must also disengage these reflected forces and substitute simulated forces generated by the simulator

3.2 Disengaging position synchrony using a Position Clutch

In some cases, we may want to move the forward simulator in space without actually sampling the path, for example to pre-position for a complex manipulation. Thus we may wish to disengage the simulator from recording any positioning commands. To do this we disengage a "Position-Clutch" that allows forward simulation without path planning. This provides a positioning-synchrony constraint relaxation analogous to the earlier time-synchrony relaxation. In this case no position information is entered into the command buffer until the position clutch is reengaged, at which time the reengagement position is entered into the control buffer, and later used by the actual robot in path interpolation from the previous path position. If the real system catches up with a position-clutch disengagement point, it hits an "empty mark" in the command buffer and must wait for further path data to enter the buffer (see Fig. 3.1).

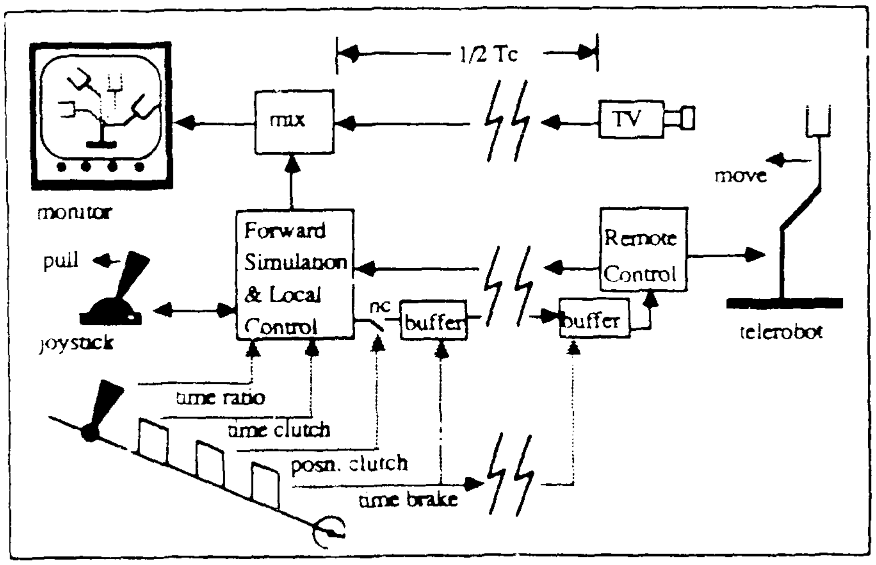


Figure 3.1: Using time and position clutches, time ratio and time brake to control forward simulation path planning in a tele-autonomous system.

Note that the time-clutch can be disengaged while the position clutch is engaged. But disengaging the position clutch overrides any actions of the time clutch. Reengaging the time-clutch after an interval of time-saving places the forward simulator and predictor display in much the same relationship to the remote unit as when operating through a time delay, with the operator directly generating a synchronized time and position trajectory in advance of the return video. In all these cases, use of time and position clutches can be superimposed over time delays in the communications between the local and remote machines (assuming adequate buffer capacity).

The command buffer can be constructed to hold more complex commands (in parallel) than just simple moves, enabling the operator to mark certain path positions as places where an embedded task is to be done. For example, suppose a switch must be pushed at some point along a path, and that the manipulation program for switch pushing resides in the remote controller. The operator might just mark the spot on the path when the forward simulator reached the switch (momentarily disengaging the time and position clutches and manipulating a screen menu entry signifying switch pushing). The telerobot manipulator (or remote vehicle, etc.) would then execute the task when it arrived at that point on the real path, i.e., when that path information emerged from the command buffer.

3.3 Scenario showing use of the Time and Position Clutches

A short scenario for using the two clutches follows: We perform a complex maneuver with clutches engaged. We then disengage the time-clutch to quickly hop over a series of simple manipulation movements, such as pushing a series of switches. A faint "smoketrail" superimposes the forward simulation path over the return video display, helping us visualize our progress along the chosen path. Having saved some time, we then disengage the position clutch, and by trial and error movements position our manipulator in simulation to be at the right place to begin a complex maneuver. During this phase, the simulation-generated manipulator image moves on the display, but leaves no "smoketrail" of a committed path. Upon reaching the correct position and orientation to begin the next maneuver, we reengage both clutches (the "smoketrail will now display the new interpolated path segment) and wait for the remote system to catch up. We then begin the maneuver. In this way we (i) save some time, (ii) use the time saved to later preposition for another action, (iii) avoid taking the actual remote system through complex, manipulatively unnecessary prepositioning movements, and (iv) do this all in a natural way through simple controls.

Note that the following of paths generated during time-clutch disengagements can be done by crude methods such as simple interpolations while keeping movements slow enough to avoid robot rate limits. Or it could be done by sophisticated methods that take into account the full dynamics of the situation and drive the remote telerobot at nearly its maximum feasible rate along the path, given specific actuator limits and desired mean-error limits. This defines a large tradeoff space in the computational complexity of trajectory generation vs the time-performance and robustness of the resulting manipulation.

3.4 Time-Ratio Control.

So far we have implied a 1:1 ratio of forward simulation time to real time when a tele-autonomous system is operated in time-synchronized mode (time clutch engaged). This needn't be the case. For example, we might be able to operate the simulator much faster than the telerobot can follow, and wish to plan the path sequence in a synchronized, but scaled, time. So instead of using the time-clutch to disengage time synchrony, we might want to establish a low time-ratio between simulated and real time. But there also might be tasks that the telerobot can do far more rapidly than we could prescribe with the simulator. In those situations, if we had "saved up" some time, we could establish a high time-ratio of simulated time to real time and slowly perform a maneuver to be later done very rapidly by the telerobot (when it catches up to that section of the path).

These "time-ratio" scalings relating real-time to simulated time can be easily implemented and then controlled by allowing a change of time-ratio while the time-clutch is disengaged (analogous to changing the gear-ratio of a vehicle while the clutch is disengaged). The time-ratio then holds its new value until changed again during a later time-clutch disengagement. Time-ratio scaling should not be confused with operating while the time clutch is disengaged (where no fixed relationship is specified between simulator time to generate a path and telerobot time to follow the path).

3.5 Handling of contingencies by using Time Brakes

What are we to do if we are forward simulating way out in front of the telemanipulator and suddenly see (in return video) something intrude into the planned path of the manipulator? To handle such simple contingencies, we introduce a mechanism we call a "Time Brake". Depression of the time brake disengages the clutches and "decelerates simulated time" by incrementally extracting (LIFO) previously generated position commands from the command buffer. The forward simulator is correspondingly moved in reverse back down the path. This allows the operator to move (as quickly as desired) back in time along the forward simulation path until located in space on the earlier side of the obstacle. We also provide an "emergency brake" that "immediately" empties the command buffer and halts the telemanipulator (subject, of course, to overshoots due to manipulator compliance and/or dynamic constraint management, and to races against 1/2 Tc).

3.6. Manipulation and cognition control-transitions and their mirrorings in telerobotic and supervised autonomous systems

The control methods described in this paper enable simple and smooth handoffs from local human teleoperation control to and from remote machine manipulation control (using downloaded manipulation commands). But they also provide a base-level protocol that enables easy mechanizations of the other types of transitions from local-control by human or machine of cognition-or-manipulation, to local-or-remote machine control of manipulation-or-cognition. Seen this way, the augmented teleoperation and autonomous systems mirror into one another to become tele-autonomous systems. Human or machine agents on "either side of the mirror" can exploit similar forward simulation and control handoff methods.

We hypothesize that human operators of this technology can learn to accomplish graceful and efficient hand-offs, rendezvous and recaptures of real-time thinking and manipulation tasks, and that human-or-machine cognition-or-manipulation operators can also exploit the forward simulation constraint relaxations to improve performance in many situations. Humans could thus supervise, or be dynamically embedded into, complex human-machine task lattices, taking and releasing control of subtasks at appropriate times and places.

For example, the tele-autonomous technology provides a framework that enables us to mimic the aircraft student-instructor scenario, with either student or instructor being machine or human, each undertaking sequences of cognition and manipulation tasks. Consider for example the situation in Figure 3.2., where we see a telerobot, R, following a path specified by a forward simulation, S, that is proceeding with its time clutch disengaged. The operator of simulator S then disengages the position clutch and moves S down and to the right. S is now essentially disengaged from any connection with the telerobot. At the same time, some other operator (human or artificial) is maneuvering another simulator, S', down towards the forward planned path (S' is also operating with its position clutch disengaged). When S' gets "close enough" to the point where S left off path planning, S' can then engage its position clutch, and take over control of the telerobot (subject to acquisition intermediation by arbiter or collision detect mechanism at the telerobot). The interaction controls can be factored from the actual manipulations throu. 'ı a small, but important, time increment. The two "players" can formulate and interact using shared visualizations and rapid cueing methods, much as skilled sports players learn to do. This simple example is suggestive of a number of more elaborate protocols and scenarios that can be constructed on top of the low-level hand-off and rendezvous protocol.

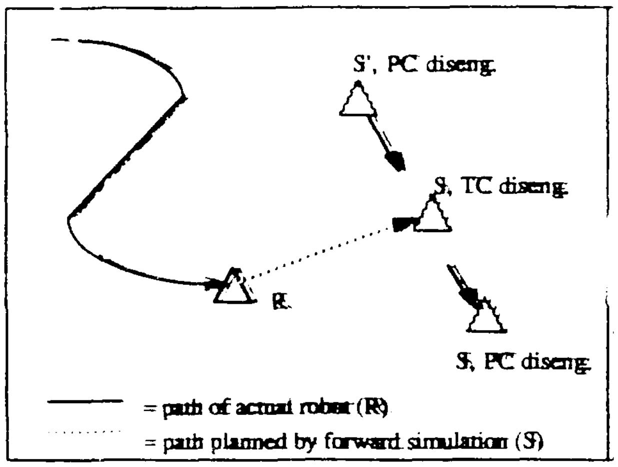

S', PC diseng.
S, TC diseng.
R
S, PC diseng.
= path of actual robot (R)
= path planned by forward simulation (S)

Figure 3.2: Using time and position clutches to "hand-off" and to "rendezvous" with a manipulation task.

4. INITIAL RESEARCH ISSUES AND HYPOTHESES

Our initial approach to tele-autonomous system research is to form hypotheses concerning the overall human-machine system much as current human-computer interaction work models unassisted human perceptual, cognitive, and motor systems. We then test these ideas by experiment. For example, Fitt's law [CAR83] predicts that the time for the eye-mind-hand task of touching an object of linear size S at a distance D is given by $T = K\log_2(D/S + 0.5)$, where $K \sim 100$ msec./bit. Therefore, simple tasks based on varying the relative sizes of objects, and the distances between objects, might produce meaningful trials of the various modes of tele-autonomous operation. Could performance operate under some sort of scaled Fitt's law in some modes? Or is it more complex than that? We could find out, and perhaps develop some insights and principles on how to best design such systems.

A simple 2-dimensional testbed can accomodate a wide range of such performance trials, such as the manipulation trial sketched in figure 4.1. In that figure, we see a number of "switches" of linear size "S" located in sequence at known positions in the manipulator workspace. Each switch is distance "D" from its predecessor in the sequence. The objective is to touch each switch in the sequence as rapidly as possible. Increasing the ratio D/S corresponds to increasing task manipulation complexity, possibly requiring more time for manipulation convergence, as abstracted in Fitt's law.

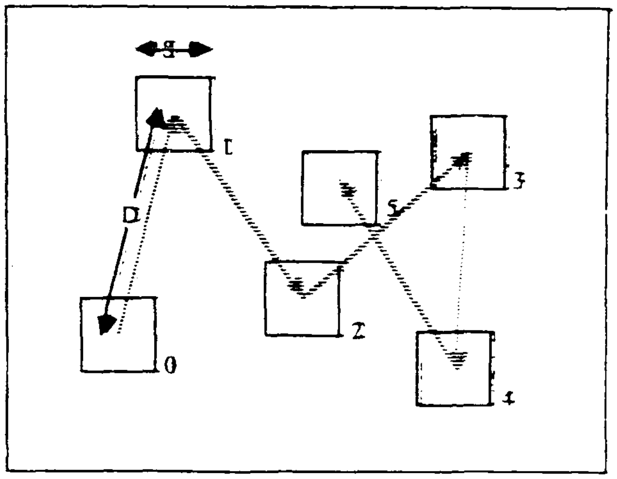


Figure 4.1: Simple testbed and example manipulation trial.

We can explore and answer many questions using this simple testbed. What is the functional form of the reduction in manipulation time, over direct teleoperation, that can be obtained when using the different augmentations of control? How are these functions and times affected in the presence of communications delays? How are the times affected by the difficulty of the manipulation targeting task (larger mean values of D/S). What are the effects of other system parameters, such as joystick force constants and robot velocity limits? What determines the percentage of the task execution time that the system operator needn't be in the control loop, so that they can be available for performing other functions?

The quantitative results of such trials can yield important early measures of the forms and dimensions of performance improvements possible with the tele-autonomous controls. The results can then help guide planning of further trials and the exploratory evolution of the technology.

5. EXPERIMENTAL ENVIRONMENT AND RESULTS

We are building a general experimental environment in which to create and evolve tele-autonomous technology. In this section we describe our initial facility and some early experiments with the different control methods. These early experiments are being done on transitions viewed from the telerobotic point of view, and provide the basis for planning later experiments in which we will study transitions from the various mixed telerobotic/autonomous systems points of view.

Our initial facility consists of a Unimation PUMA 560 used as a telerobot, controlled by either a DEC VAX 11/750 computer or an Apollo DSP90 computer (both modes are available). A force and moment sensitive joystick is used as an input device to provide a rate input for the telerobot. To simulate a variety of real robots typical of those used in space or undersea operations, appropriate velocity limits are placed on each joint. To simulate remoteness of the telerobot from the operator, a variable delay can be inserted between a forward simulator generated control stream and the PUMA, using buffer to hold the trajectory sample stream.

A high-performance Silicon Graphics IRIS workstation is used to generate and mix the display of the forward simulator and the telerobot, with the telerobot seen either in return video or as a graphics model (the latter can be useful since it is both easier to obtain correspondence between the overlay and the simulation and easier to modify the viewpoint of the system).

Simple time and position clutches have been implemented in the system. The logical operation of the clutches is portrayed as a state diagram in Figure 5.1., which shows the allowable combinations of time and/or position synchrony and the transitions between them. There are three allowable control states from the teleoperation point of view: 1) TSyn & PSyn – time and position synchrony, 2) PSyn – position synchronism only, and 3) NoSyn – both time and position synchrony disengaged. Changes between these states are controlled by switches that operators push with their feet – similar to the clutch in an automobile. The joystick moves the simulator in all three states, but the state determines the effect of simulator movement on path planning and path buffer encoding.

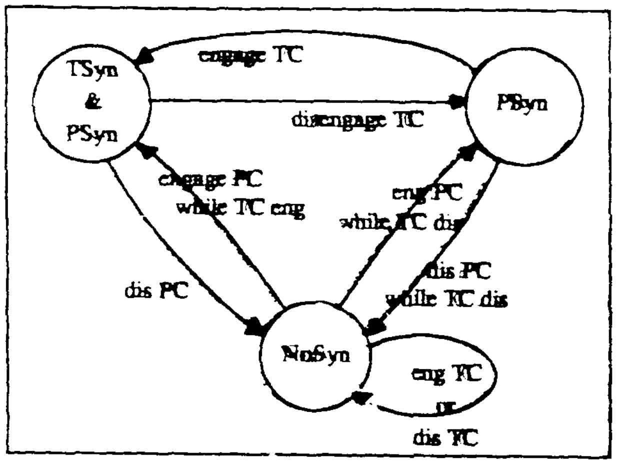


Figure 5.1: Diagram of forward simulator states and the transitions caused by time and position clutches.

A foot operated time brake is implemented which when pressed (i) disengages both clutches, causing an overriding transition to NoSyn, and (ii) begins deleting entries in the command buffer (LIFO), thus running the forward simulator back down the previously generated path.

In the time and position synchronized state (TSyn & PSyn), the force and moment outputs of the joystick are sampled at the input rate required by the PUMA. The forces and moments obtained are treated as vectors of desired velocities in Cartesian space. These are integrated to obtain position samples, and these samples are placed in the command buffer. While in time synchronized mode the buffer is emptied at the same rate it is filled, and the values obtained from the buffer are input to the PUMA system, which treats each sample as a goal to reach in its sample period, by slewing any or all of its six joints. When we wish to simulate situations where the actions can be visualized and simulated much faster than they can be manipulated, such as when moving large structures in space [NAS81] or underwater, we place a selected angular velocity limit $Wj(i) < Wjmax(i)$ on each axis, i, of the PUMA.

When the time clutch is disengaged, and the position syncronized (PSyn) state is entered, the robot-model physical constraints on simulation distance covered per time-sample are removed, and the usual joystick force and torque constants are multiplied by a gain constant, Gs, enabling the operator to rapidly slew the simulator. Path samples may be generated at varyingly wider path intervals than is possible when control is synchronized in time. Values removed from the buffer may thus request incremental moves larger than can be accomplished in one PUMA sample period, given selected constraints on the angular velocities of the PUMA joints. When this occurs, the commanded move is interpolated and spread across more than one PUMA sampling interval, with the actual PUMA rate of motion constrained as above by the selected joint angular velocity constraints. Thus, the simulated telerobot can be moved out along a path well ahead of the real telerobot, and we can "save up some time". In addition, the real robot follows this path at nearly its maximum rate, for a given set of values of Wjmax (i). Thus we predict that the overall manipulation time, Tm, will usually be smaller using this mode than if there were a fixed ratio between simulation time and real-time (as for example in time-ratio control).

If a transition is made from PSyn to the unsynchronized (NoSyn) state, the operator becomes free to move the simulated telerobot without values being placed in the command buffer. Then, when a transition is made back to PSyn, the current position of the simulated telerobot is placed in the buffer. When this value is extracted from the buffer, the telerobot makes interpolated incremental moves directly towards that desired position, without going through all of the motions the operator had to use to get to that position. This enables "saved up" time to be used to "edit" out some real time and path motions used, for example, for complex prepositioning.

5.1. Experimental Parameters, Trials and Results

In our first trials we used simple, random, 2-dimensional, 5-switch testbeds similar to that in Figure 4.2. We conducted a series of trials varying the following parameters:

(i) Three different subjects (X, Y, Z) each performed a series of manipulation tasks using the testbed. Two times were recorded for each trial: The subject's time to specify the manipulation, (Ts), and the system's time to complete the manipulation. (Tm). We also recorded the actual manipulation path length, Lm. The ratio of Lm to the minimum path (~ 5D) provides a measure of one dimension of operator skill).

(ii) The series of switch-touching tasks varied from simple to difficult by ranging from low values of D/S ~ high values of D/S (D = 500 mm.; S = 25, 50, 75 [illegible]).

(iii) Communication delays, Tc, of 0, [illegible] sec. were used.

(iv) Tasks over the range of difficulty and the range of communication delays were performed by each subject using: (a) direct teleoperation (TOP), (b) teleoperation assisted by forward simulation (TOP+FS), and (c) teleoperation assisted by forward simulation and time clutching (TOP+FS+TC).

During these first trials, other key system parameters were held constant as follows:

(i) Workspace to monitor-screen length-ratio = 8:1.

(ii) Joystick sample period = 0.017 sec.

(iii) Joystick force constant = 0.01 mm per oz per sample period = 0.6 mm per sec. per oz.

(iv) Joystick torque constant = 0.0012 rad. per oz-in per sec.

(v) Joystick gain constant, Gs, in (TOP+FS+TC) = 4.0.

(vi) Angular velocities of all 6 PUMA joints were limited to Wj < Wjmax = 0.5 radians per sec. (but see also below).

Other comments on our methods: The chosen constant values yield moderately responsive controls when moderate joystick forces and torques are applied. The angular velocity limits yield a moderately fast robot (slower than the PUMA can go at its fastest, but very, very much faster than a scaled shuttle arm). All subjects engaged in preliminary learning trials. All used the joystick "one-handed". Trials began after a period of preliminary learning. Comparable power-law of practice performance levels [CAR83] were recorded for each mode.

Results of some of these initial trials for one subject are plotted in Figures 5.2, which shows the specification time (Ts) and manipulation times (Tm) for tasks over the range of D/S difficulty holding D = 500mm. Included are results for communication delays, Tc, of 0.0, 2.0 and 4.0 seconds. The results are displayed for the three relevant modalities of control: (a) TOP, (b) (TOP+FS), and (c) (TOP+FS+TC).

We note that a comparison of TOP and (TOP+FS) repeats experiments of Sheridan, et. al. [HAS86, SHE86], confirming the results of that work. We see that (TOP+FS) gives a significant gain in both Ts and Tm over TOP alone. Then we notice that (TOP+FS+TC) gives another significant gain in Ts over (TOP+FS). In the initial trials, we found that Wjmax = 1.0 rad./sec. was high enough for the robot's Tm time to keep up with even the shortest (TOP+FS+TC) Ts times (see Fig. 5.2). We then found that Wjmax = 0.5 rad./sec. constrained Tm so that subjects could easily outpace the robot and save up time (see Fig. 5.2). Many of the initially hypothesized forms of results were demonstrated using these parameter ranges.

We then noticed that Ts and Tm grew less rapidly in D/S than anticipated. We hypothesized that D = 500mm was large enough, given the joystick constants and Wj values, to produce dynamic constraints related to D and not just D/S. So we repeated scaled versions of these trials at smaller values of D.

Figure 5.3 shows the results for D = 250 mm and S = 50, 37.5, 25 and 12.5 mm. (with work to screen scale = 16:1, and Wjmax = 0.5). It also includes results using D = 125mm and S = 25, 18.7, 12.5 and 6.2 mm (with work to screen scale = 32:1, and Wjmax = 0.5). These results are interesting, because for all three modes the data per mode at D = 250mm and D = 125 mm essentially fall on top of one another. The 250mm and 125mm curves for each mode lie well below those for D = 500mm. Refer Fig. 5.2 for the time-clutch mode data for D = 500mm (it would partly overly the Fig. 5.3 time-clutch data).

At this scale the system operates in a "Fitt's law-like" region, with Ts and Tm being functions of D/S (but not D), with the values in most cases at D/S = 20 about twice those at D/S = 5. For Wjmax = 0.5, the robot's Tm at this scale could stay up with the subjects Ts. We varied Wjmax and found values of 0.35 (for D = 250) and 0.25 (for D = 125) that yielded demos of significant time differences between Tm and Ts for (TOP+FS+TC) mode on the easier tasks (see Fig. 5.3).

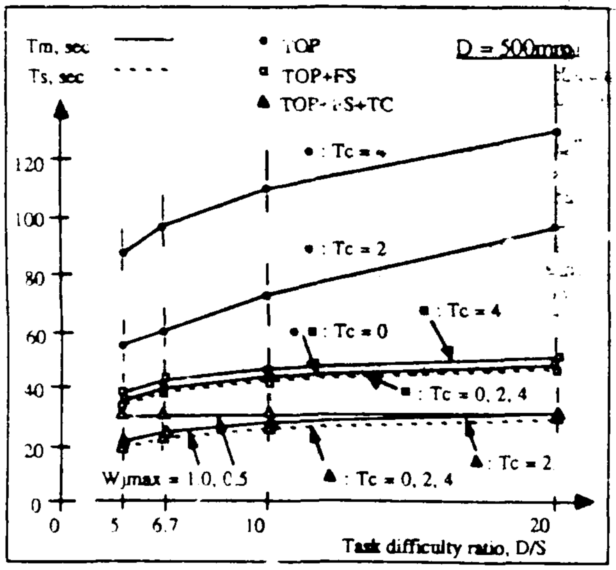


Figure 5.2: Initial trial results, showing Ts, Tm as functions of system and task parameters for three modes of control.

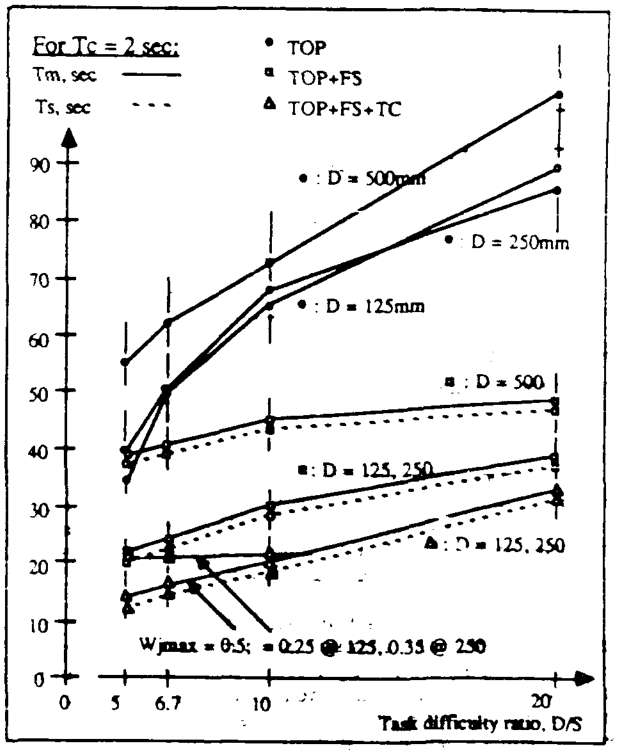


Figure 5.3: Trials showing Ts, Tm as functions of system and task parameters, for several values of task size scale D.

On further scaling-down of D, the system enters "the Heisenberg" region on the harder tasks (S < 2 to 3mm Position sample-sizes, interpolator discretization and operator jitters cause large increases and variances in Tm and Ts (like trying to poke at things with a needle under a microscope).

Throughout the trials, subjects noticed striking difference in the "feel" of the different control modes, and develope special tactics for coping with each mode. Most treated TOP in the presence of delays like hitting a series of "successively shorter golf shots", trying to get closer each time. Subjects controlled (TOP+FS) aggressively, firmly driving the simulator to each switch. The (TOP+FS+TC) mode was usually handled with finesse, so as to drive it fast, but not so fast as to yield a wild path and thus large Tm and large Lm/5D.

In addition to these preliminary quantitative results, we have demonstrated the use of the position clutch to enable graceful handoffs of control by one agent and rendevous of control by another agent. This is done by simply having two human operators swap use of the controls following disengagement of the position clutch once the forward simulation is out well ahead of the telerobot. We have compiled a video report showing the above experiments, demonstrations and control effects [CON87].

5.2. Plans for further experiments and concept demonstrations

We are continuing the above series of trials, varying additional system parameters. We are also preparing additional types of experiments and demos. The time brake and position clutch will be used to determine their effects on specification and manipulation times, and manipulation path lengths (Skilled operators can use the time brake to "erase" poor path sections, and the position clutch to make and "jump across" gaps during overshoots). The time brake will be tested in contingencies (example: an obstacle falls across the planned path behind the forward simulator). Transitions involving cognitive/manipulative task-nesting will be explored. We will add the command buffer and interpolator modifications,and HCI controls, to implement and demonstrate time-ratio control during time-synchronized forward simulation.

We also plan to attempt demos of simple forms of role-reversal by having the tele-automaton do the path planning, and letting the human rendezvous to telemanipulate along selected portions of the projected path. In the role reversal demo a route planner uses AI techniques to plan a path through a maze. The planner then places the path into the forward simulator and, when necessary, calls upon the human to take over and manipulate through certain path segments. The human then just drives the manipulator along the displayed path segment. This mimics a human taking over the driving of an ALV while the ALV remains under machine cognitive control. This environment will also enable demos of human intervention in cognitive tasks, for example to assist in planning the route if the machine gets stuck in that high-level planning task.

We found basic principles such as Fitt's law very useful in thinking about forms of testbeds and hypotheses for our early trials. We need to consider additional system parameters and also the dynamics of the human/machine combination, generate further hypotheses regarding factors affecting performance, and then design experiments to test these ideas. Such work may eventually produce principles and design rules for tele-autonomous manipulation systems.

6 FUTURE RESEARCH CHALLENGES

This initial tele-automation work suggests opportunities for coordination of research in several specialized fields. It also raises issues concerning research equipment infrastructure.

Tele-autonomous technology presents new challenges in human computer interaction. We have proposed a set of interface controls that are conceptually simple and easy to mechanize. The controls are generic ones that may be applicable in many different specialized situations. They are also cognitively and manipulatively accessible to the uninitiated by analogy. But many other new human interface aspects haven't been pinned down at all. How is the operator to visualize where they are, who has control of what, and who they give control to next as they enter or leave some subtask within a complex task lattice? What measures can we provide concerning operator performance, and what feedback can we provide? And what about the analysis and design of cognitive and manipulation tasks themselves? Research can perhaps provide better measures of joint human-machine cognitive-manipulative performance. Analyses similar to those in [CAR83] may then lead us to design intermixings of human and machine activity that yield substantial improvements in overall performance.

Research challenges arise in robotics, such as the eventual need to perceive, model and forward simulate not only the remote tele-automaton, but also portions of the remote environment itself. Forward simulation will work fine when interacting with static objects, but what about interactions with moving objects? Even if we knew how to specify interactions with moving objects, such work would be severely constrained by the high computational complexity of present methods for representing and simulating mechanical systems. Further basic work, such as that of Hopcroft, on efficient representation and simulation of mechanical systems is required if we are to handle problems of really interesting complexity [HOP87].

More work is needed on methods for path-error specification and associated methods for the time optimization of path following, such as in [SUH87]. Additional work is also needed on autonomous "reflex" actions that the remote robot can perform when encountering uncertainties (particularly those involving contact) not modelled in the forward simulation. We also need augmented AI programming environments that interface in such a way with real-time programming environments as to easily enable rapid estimation of time available for short-term AI planning tasks (enabling us to select among AI methods as a function of available time).

We believe that fundamental work can be done in these areas with modest robotic laboratory equipment. AI techniques [WIN84] and expert system technology [HAY83] have matured so that roboticists can now mechanize knowledge-intensive cognitive functions well beyond their reach just a few years ago, and can run these systems on accessible workstations. Thus mixings of manipulation and cognition technologies are now ripe for research exploration.

However, some experiments will benefit from multi-dimensional teleoperators or high-tech automation or autonomous system technology. One way to gain access to such expensive equipment is to treat remoteness as a feature: For example, we are negotiating connection of our tele-autonomous control equipment via satellite links with automation systems at several remote sites. As such efforts provide useful testbeds, others might exploit shared access to the same remote facilities. Shared access to capital equipment has obvious costs benefits, but in addition can stimulate standards, collaborations and healthy direct competitions among researchers. Shared access to silicon foundries greatly increased the productivity of the VLSI research community [CON81]; a remote tele-automation facility could play an analogous role in tele-autonomous systems research.

7. SUMMARY

We have introduced basic functional concepts for tele-autonomous technology and an architectural framework for implementing the technology, using controls over time and position synchrony that enable simple structuring of control transitions. We have proposed hypotheses concerning capabilities of the technology, described our environment for investigating these phenomena, and discussed the results of early tests of some of the hypotheses. The results indicate how telerobotics can be extended to include projection of cognitive activity and autonomous systems extended to accomodate smooth transitions of cognitive or manipulative responsibility between machine and human operator. Through such extensions, the two technologies merge into one of "tele-autonomous systems". Finally, we have also sketched some further lines of research suggested by this initial work.

We believe that tele-autonomous systems research can yield methods and systems for improved projection of intelligent, manipulative action at a distance in time and space. This interdiscipline presents interesting new research opportunities to teams having expertise in robotics and automation, artificial intelligence, and the psychology of human-computer interaction. We envision many possible applications for the resulting technology, not only in space and defense systems, but also in design and production systems, and eventually in personal and recreational environments.

8. ACKNOWLEDGEMENTS

We gratefully acknowledge support of the Center for Machine Intelligence and the College of Engineering of the University of Michigan, the support provided by the grant of a high performance IRIS graphics workstation from Silicon Graphics, Inc., and the contributions of Lee Hagerhorst and Lejun Shao to the rapid prototyping of our first experimental systems.

9. REFERENCES

[CAR83] Card, S. K., Moran, T. P. and Newell, A., *The Psychology of Human-Computer Interaction*, Lawrence Elbaum Assoc., Hillsdale, NJ, 1983.

[CON81] Conway, L., "The MPC Adventures: Experiences with the Generation of VLSI Design and Implementation Methodologies", *Second Caltech Conference on VLSI*, California Institute of Technology, Jan. 1981.

[CON87] Conway, L., Volz, R. and Walker, M., "New Concepts in Tele-Autonomous Systems," Univ. of Michigan Robotics Laboratory Video-Report, Feb. 1987.

[DAR83] DARPA, *Strategic Computing New-Generation Computing Technology - A Strategic Plan for its Development and Application to Critical Problems in Defense*, Defense Advanced Research Projects Agency, Arlington, VA, October 28, 1983.

[DAV85] Davis, D. B., "Assessing the Strategic Computing Initiative", *High Technology*, April 1985.

[HAS86] Hashimoto, T. and Sheridan, T. B., Paper to be published in the Japanese Journal of Ergonomics.

[HAY83] Hayes-Roth, F., Waterman, D. W., Lenat, D. B., ed., *Building Expert Systems*, Addison-Wesley, Reading, MA, 1983.

[HOP87] Hopcroft, J and Krafft, D., "The Challenge of Robotics for Computer Science." In *Advances in Robotics*, Vol. 1, *Algorithmic & Geometric Aspects of Robotics*. C. Yap and J. Schwartz, ed., Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[MAR86] Martin Marietta Corporation, "The Autonomous Land Vehicle 2nd Quarterly Review", DARPA ALV Workshop, June 23-27, 1986.

[NAS81] NASA, Shuttle Flight Operations Manual, Payload Deployment and Retrieval Systems," Vol. 16; Flight Operations Directorate, Johnson Space Center, June 1, 1981.

[NOY84] Noyes, M. and Sheridan, T. B., "A Novel Predictor for Telemanipulation through a Time Delay", *Proc. of the Annual Conference on Manual Control*, NASA Ames Research Center, Moffett Field, CA, 1984.

[RTI82] RTI, *RTI Force Sensing Wrist User's Manual*, Robot Technology, Inc., Los Altos, CA, 1982.

[SHE86] Sheridan, T. B., "Human Supervisory Control of Robot Systems", *Proc. of the IEEE International Robotics Conference*, April 1986, pp. 808-812.

[SUH87] Suh, S. H. and Bishop, A. B., "Tube Concept and Its Application to the Obstacle Avoidance Minimum-Time Trajectory Planning Problem." Univ. of Michigan Robotics Laboratory paper submitted to the *IEEE Journal of Robotics and Automation*.

[VER86] Vertut, J. and Coiffet, P., "Teleoperations and Robotics: Applications and Technology," *Robot Technology*, Vol. 3B, English Trans., Prentice-Hall, 1986.

[WIN84] Winston, P. H., *Artificial Intelligence*, 2nd Ed., Addison-Wesley, Reading, MA, 1984.

AIAA-87-1687
Software Architecture For Manufacturing And Space Robotics

J. S. Albus, R. Lumia and H. McCain,
National Bureau of Standards,
Gaithersburg, MD

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

March 9-11, 1987/Arlington, VA

SOFTWARE ARCHITECTURE FOR MANUFACTURING AND SPACE ROBOTICS

J.S. Albus, R. Lumia, and H. McCain
Robot Systems Division
Metrology Building, Room B-124
National Bureau of Standards
Gaithersburg, MD 20899

ABSTRACT

A hierarchical architecture is described which supports space station telerobots in a variety of modes. The system is divided into three hierarchies: task decomposition, world model, and sensory processing. Goals at each level of the task decomposition hierarchy are divided both spatially and temporally into simpler commands for the next lower level. This decomposition is repeated until, at the lowest level, the drive signals to the robot actuators are generated. To accomplish its goals, task decomposition modules must often use information stored in the world model. The purpose of the sensory system is to update the world model as rapidly as possible to keep the model in registration with the physical world. This paper describes the architecture of the entire control system hierarchy and how it can be applied to space telerobot applications.

1. INTRODUCTION

One of the major directions on which the robot research community has concentrated its efforts is concerned with planning and controlling motion. Given a specific task, a motion plan must be calculated which meets the task requirements. Then, the plan must be executed; there must be sufficient control for the robot to adequately effect the desired motion.

Trajectories are often planned as straight lines in Cartesian space [1]. Whitney [2,3] developed the resolved motion rate control method for Cartesian straight line motions. Paul [4,5,6] used homogeneous coordinate transformations to describe a trajectory as a function of time, and Taylor [7] used coordinated joint control over small segments to keep the trajectory within a specified deviation of the desired straight line trajectory.

While the research described above employs a "kinematic" approach to robot control, another direction of research takes the manipulator "dynamics" into account in the description of robot motion. The dynamic equations of motion are described either by the Lagrangian formulation [8] or by the Newton-Euler equations [9]. Algorithms and computer architectures have been suggested which promise real-time dynamic robot control [10,11].

Another aspect of motion control is concerned with the variables being controlled. The research described to this point was concerned primarily with position control. The robot moved from an initial position to a goal position. While this is perhaps the most common mode, there are many applications for robots which suggest that other variables should be controlled. For example, force control would be desired for assembly operations. Raibert and Craig [12] suggest a method for hybrid position/force control of manipulators.

These examples point to the more general problem of sensory processing. For a great deal of robot motion research, sensory processing has been limited to joint positions, velocities, and accelerations. However, other sensors are often required to accomplish tasks. The control community has concentrated on the control aspects of the robot and as result, little emphasis has been placed on sophisticated sensory processing.

Machine vision, an offshoot of image processing research, has recently been associated with advanced robot applications. One of the most interesting directions in this research area is concerned with sensor controlled robots. Operating with the constraints imposed by real-time robot control, early methods used structured light and binary images [13,14,15,16]. These approaches, though developed at different institutions, shared many concepts. One of the important subsequent research efforts went toward the development of model-based image processing. Bolles and Cain [17] used models of objects to guide the algorithms in a hypothesis/verification scheme known as the local feature focus method. The concept has recently been extended from two dimensional (i.e. nearly flat) objects to three dimensional objects [18]. Although the approaches described here have led to a better understanding of real-time vision processing, the systems lacked a sophisticated interconnection with the robot control system.

The Automated Manufacturing Research Facility (AMRF), developed at the National Bureau of Standards, is a hierarchically organized small-batch metal machining shop [19]. It separates sensory processing and robot control by a sophisticated world model. The world model has three complementary data representations. Lumia [20] describes the CAD-like section of the model. Shneier, Kent, and Mansbach [21] describe the octree and table

[illegible]

representations supported by the model. The model generates hypotheses for the features which are either verified or refuted by empirical evidence. The sensory system's task is to update the appropriate parts of the world model with new or revised data as rapidly as possible. The control system accesses the world model as desired to obtain the current best guess concerning any aspect of the world. Shneier, Lumia, and Kent [22] describe the sensory system and its operation in greater detail. The AMRF was the first deliberate attempt to tie together sensory processing, world modeling, and robot control in a generic fashion. The system developed for the AMRF is applicable to more than manufacturing. This paper describes its use in space telerobotics.

2. A FUNCTIONAL SYSTEM ARCHITECTURE

The fundamental paradigm is shown in Figure 1. The control system architecture is a three legged hierarchy of computing modules, serviced by a communications system and a common memory. The task decomposition modules perform real-time planning and task monitoring functions, and decompose task goals both spatially and temporally. The sensory processing modules filter, correlate, detect, and integrate sensory information over both space and time in order to recognize and measure patterns, features, objects, events, and relationships in the external world. The world modeling modules answer queries, make predictions, and compute evaluation functions on the state space defined by the information stored in common memory. Common memory is a global database which contains the system's best estimate of the state of the external world. The world modeling modules keep the common memory database current and consistent.

2.1. Task Decomposition - H modules (Plan, Execute)

The first leg of the hierarchy consists of task decomposition H modules which plan and execute the decomposition of high level goals into low level actions. Task decomposition involves both a temporal decomposition (into sequential actions along the time line) and a spatial decomposition (into concurrent actions by different subsystems). Each H module at each level consists of a job assignment manager JA, a set of planners PL(i), and a set of executors EX(i). These decompose the input task into both spatially and temporally distinct subtasks as shown in Figure 2. This will be described in greater detail in section 4.

2.2. World Modeling - M modules (Remember, Estimate, Predict, Evaluate)

The second leg of the hierarchy consists of world modeling M modules which model (i.e. remember, estimate, predict) and evaluate the state of the world. The "world model" is the system's best estimate and evaluation of the history, current state, and possible future states of the world, including the states of the system being controlled. The "world model" includes both the M modules and a knowledge base stored in a common memory database where state variables, maps, lists of objects and events, and attributes of objects and events are maintained. By this definition, the world model corresponds to what is widely known throughout the artificial intelligence community as a "blackboard" [23]. The world model performs the following functions:

1. Maintain the common memory knowledge base by accepting information from the sensory system.

2. Provide predictions of expected sensory input to the corresponding G modules, based on the state of the task and estimates of the external world.

3. Answer "What is?" questions asked by the executors in the corresponding level H modules. The task executor can request the values of any system variable.

4. Answer "What if?" questions asked by the planners in the corresponding level H modules. The M modules predict the results of hypothesized actions.

2.3. Sensory Processing - G modules (Filter, Integrate, Detect, Measure)

The third leg of the hierarchy consists of sensory processing G modules. These recognize patterns, detect events, and filter and integrate sensory information over space and time. The G modules at each level compare world model predictions with sensory observations and compute correlation and difference functions. These are integrated over time and space so as to fuse sensory information from multiple sources over extended time intervals. Newly detected or recognized events, objects, and relationships are entered by the M modules into the world model common memory database, and objects or relationships perceived to no longer exist are removed. The G modules also contain functions which can compute confidence factors and probabilities of recognized events, and statistical estimates of stochastic state variable values.

2.4. Operator Interfaces (Control, Observe, Define Goals, Indicate Objects)

The control architecture defined here has an operator interface at each level in the hierarchy. The operator interface provides a means by which human operators, either in the space station or on the ground, can observe and supervise the telerobot. Each level of the task decomposition hierarchy provides an interface where the human operator can assume control. The task commands into any level can be derived either from the higher level H module, or from the operator interface. Using a variety of input devices such as a joystick, mouse, trackball, light pen, keyboard, voice input, etc., a human operator can enter the control hierarchy at any level, at any time of his choosing, to monitor a process, to insert information, to interrupt automatic operation and take control of the task being performed, or to apply human intelligence to sensory processing or world modeling functions.

The sharing of command input between human and autonomous control need not be all or none. It is possible in many cases for the human and the automatic controllers to simultaneously share control of a telerobot system. For example a human might control the orientation of a camera while the robot automatically translates the same camera through space.

2.4.1 Operator Control interface levels

The operator can enter the hierarchy at any level. The operator control interface interprets teleoperation in the fullest sense: a teleoperator is any device which is controlled by a human from a remote location. While the master-slave paradigm is certainly a type of teleoperation, it does not constitute the only form of man-machine interaction. At different levels of the hierarchy, the interface device for the human may change but the fundamental concept of teleoperation is still preserved. Table 1 illustrates the interaction an operator may have at each level.

The operator control interface thus provides mechanisms for entering new instructions or programs into the various control modules. This can be used on-line for real-time supervisory control, or in a background mode for altering autonomous telerobot plans before autonomous execution reaches that part of the plan.

2.4.2 Operator monitoring interfaces

The operator interfaces allow the human the option of simply monitoring any level. Windows into the common memory knowledge base permit viewing of maps of service bay layout, geometric descriptions and mechanical and electrical configurations of satellites, lists of recognized objects and events, object parameters, and state variables such as positions, velocities, forces, confidence levels, tolerances, traces of past history, plans for future actions, and current priorities and utility function values. These may be displayed in graphical form, for example using dials or bar graphs for scalar variables, shaded graphics for object geometry, and a variety of map displays for spatial occupancy.

2.4.3 Sensory processing/world modeling interfaces

The operator interface may also permit interaction with the sensory processing and/or world modeling modules. For example, an operator using a video monitor with a graphics overlay and a light pen or joystick might provide human interpretative assistance to the vision/world modeling system. The operator might interactively assist the model matching algorithms by indicating with a light pen which features in the image (e.g. edges, corners) correspond to those in a stored model. Alternatively, an operator could use a joystick to line up a wireframe model with a TV image, either in 2-D or 3-D. The operator might either move the wireframe model so as to line up with the image, or move the camera position so as to line up the image with the model. Once the alignment was nearly correct, the operator could allow automatic matching algorithms to complete the match, and track future movements of the image.

2.5. Common Memory

2.5.1. Communications

One of the primary functions of common memory is to facilitate communications between modules. Communications within the control hierarchy is supported by a common memory in which state variables are globally defined.

Each module in the sensory processing, world modeling, and task decomposition hierarchies reads inputs from, and writes outputs to, the common memory. Thus each module needs only to know where in common memory its input variables are stored, and where in common memory it should write its output variables. The data structures in the common memory then define the interfaces between the G, M, and H modules.

The operator interfaces also interact with the system through common memory. The operator displays simply read the variables they need from the locations in common memory. If the operator wishes to take control of the system, he simply writes command variables to the appropriate locations in common memory. The control modules that read from those locations need not know whether their input commands derived from a human operator, or from the next higher level in the autonomous control hierarchy.

2.5.2 State Variables

The state variables in common memory are the system's best estimate of the state of the world, including both the external environment and the internal state of the H, M, and G modules. Data in common memory are available to all modules at all levels of the control system.

The knowledge base in the common memory consists of three elements: maps which describe the spatial occupancy of the world, object-attribute linked lists, and state variables.

3. LEVELS IN THE CONTROL HIERARCHY

The control system architecture described here for the Flight Telerobot System is a six level hierarchy as shown in Figure 3. At each level in this hierarchy a fundamental transformation is performed on the task.

Level 1 transforms coordinates from a convenient coordinate frame into joint coordinates. This level also servos joint positions, velocities, and forces.

Level 2 computes inertial dynamics, and generates smooth trajectories in a convenient coordinate frame.

Level 3 decomposes elementary move commands (E-moves) into strings of intermediate poses. E-moves are typically defined in terms of motion of the subsystem being controlled (i.e., transporter, manipulator, camera platform, etc.) through a space defined by a convenient coordinate system. E-move commands may consist of symbolic names of elementary movements, or may be expressed as keyframe descriptions of desired relationships to be achieved between system state variables. E-moves are decomposed into strings of intermediate poses which define motion pathways that have been checked for clearance with potential obstacles, and which avoid kinematic singularities.

Level 4 decomposes object task commands specified in terms of actions performed on objects into sequences of E-moves defined in terms of manipulator motions. Object tasks typically define actions to be performed by a single multiarmed telerobot system on one object at a time. Tasks defined in terms of actions on objects are decomposed into sequences of E-moves defined in terms of manipulator or vehicle subsystem motions. This decomposition checks to assure that there exist motion freeways clear of obstacles between keyframe poses, and schedules coordinated activity of telerobot subsystems, such as the transporter, dual arm manipulators, multifingered grippers, and camera arms.

Level 5 decomposes actions to be performed on batches of parts into tasks performed on individual objects. It schedules the actions of one or more telerobot systems to coordinate with other machines and systems operating in the immediate vicinity. For example, Level 5 decomposes service bay action schedules into sequences of object task commands to various telerobot servicers, astronauts, and automatic berthing mechanisms. Service bay actions are typically specified in terms of servicing operations to be performed by all the systems (mechanical and human) in a service bay on a whole satellite. This decomposition typically assigns servicing tasks to various telerobot systems, and schedules servicing tasks so as to maximize the effectiveness of the service bay resources.

Level 6 decomposes the satellite servicing mission plan into service bay action commands. Mission plans are typically specified in terms of satellite servicing priorities, requirements, constraints, and mission time line. The level 6 decomposition typically assigns satellites to service bays, sets priorities for service bay activities, generates requirements for spare parts and tool kits, and schedules the activities of the service bays so as to maximize the effectiveness of the satellite servicing mission. To a large extent the level 6 mission plans will be generated off line on the ground, either by human mission planners, or by automatic or semiautomatic mission planning methods.

4. DETAILED STRUCTURE OF THE H MODULES

The H module at each level consists of three parts as shown in Figure 4: a job assignment manager JA, one or more planners PL(s), and one or more executors EX(s).

The job assignment manager JA is responsible for partitioning the task command TC into s spatially or logically distinct jobs to be performed by s physically distinct planner/executor mechanisms. At the upper levels the job assignment module may also assign physical resources against task elements. The output of the job assignment manager is a set of job commands JC(s), s=1, 2, or logically, distinct jobs.

For each of these job commands JC(s), there exists a planner PL(s) and a executor EX(s). Each planner PL(s) is responsible for decomposing its job command JC(s) into a temporal sequence of planned subtasks PST(s,tt). Planning typically requires evaluation of alternative hypothetical sequences of planned subtasks. The planner hypothesizes some action or series of actions, the world model predicts the results of the action(s) and computes some evaluation function EF(s,tt) on the predicted resulting state of the world. The hypothetical sequence of actions producing the best evaluation function EF(s,tt)max is then selected as the plan PST(s,tt) to be executed by the executor EX(s).

PST(s,tt) = PL(s) [JC(s),EF(s,tt)max]

where tt is the time sequence index for steps in the plan. tt may also be defined as a running temporal index in planning space, tt = 1, 2, ..., th where th is the value of the tt index at the planning horizon. The planning horizon is defined as the period into the future over which a plan is prepared. Each level of the hierarchy has a planning horizon of one or two expected input task time durations.

Each executor EX(s) is responsible for successfully executing the plan PST(s,tt) prepared by its respective planner PL(s). If all the subtasks in the plan PST(s,tt) are successfully executed, then the goal of the original task will be achieved. The executor operates by selecting a subtask from the current queue of planned subtasks and outputting a subcommand STX(s,t) to the appropriate subordinate H module at time t. The EX(s) module monitors its feedback FB(s,t) input in order to servo its output STX(s,t) to the desired subtask activity.

STX(s,t+n) = EX(s) [PST(s,t),FB(s,t)]

where n = the number of state clock periods required to compute the function EX(s). n typically equals 1. The feedback FB(s,t) also carries timing and subgoal event information for coordination of output between executors at the same level. When the executor detects a subgoal event, it selects the next planned subtask from the queue.

Executor output STX(s,t) also contains requests for information from the world model M module, and status reports to the next higher (i+1) level in the H module hierarchy. The feedback FB(s,t) contains status reports from the H module at the i-1 th level indicating progress on its current task.

5. CONCLUSION

This paper has described a hierarchically organized control system and has shown how this generic system can be applied to telerobotic applications in space by considering the requirements of a flight telerobotic servicer for the space station.

REFERENCES

[1] M. Brady, et.al., ed. Robot Motion: Planning and Control, (Cambridge, MIT Press, 1982).

[2] D.E. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," IEEE Trans. Man-Machine Systems MMS-10, 1969, p. 47.

[3] D.E. Whitney, "The Mathematics of Coordinated Control of Prostheses and Manipulators," Journal of Dynamic Systems, Measurement, Control, Dec. 1972, p 303.

[4] R.P. Paul, "Manipulator Path Control," IEEE Int. Conf. on Cybernetics and Society, New York, p. 147.

[5] R.P. Paul, "Manipulator Cartesian Path Control," IEEE Trans. Systems, Man, Cybernetics SMC-9, 1979, p. 702.

[6] R.P. Paul, Robot Manipulators: Mathematics, Programming, and Control, (Cambridge, MIT Press, 1981.)

[7] R. H. Taylor, "Planning and Execution of Straight-line Manipulator Trajectories," IBM J. Research and Development 23 1979, p. 424.

[8] J.M. Hollerbach, "A Recursive Formulation of Lagrangian Manipulator Dynamics," IEEE Trans. Systems, Man, Cybernetics SMC-10, 11, 1980, p. [illegible].

[9] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul, "On-line Computational Scheme for Mechanical Manipulators," J. Dynamic Systems, Measurement, Control, 102, 1980, p. 69.

[10] C.S.G. Lee, P.R. Chang, "Efficient Parallel Algorithm for Robot Inverse Dynamics Computation," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-16, No. 4, July/August 1986, p. 532.

[11] E.E. Binder, J.H. Herzog, "Distributed Computer Architecture and Fast Parallel Algorithm in Real-Time Robot Control," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-16, No. 4, July/August 1986, p. 543.

[12] M.H. Raibert and J.J. Craig, "Hybrid position/force control of manipulators," J. Dynamic Systems, Measurement, Control, June, 1981, p. 126.

[13] W.A. Perkins, "A Model Based Vision System for Industrial Parts," IEEE Trans. on Computers, Vol. C-27, 1978, p. 126.

[14] G.L. Gleason, G.J. Agin, "A Modular Vision System for Sensor-controlled Manipulation and Inspection," Proc. 9th Int. Symposium on Industrial Robots, 1979, p. 57.

[15] M.R. Ward, et.al., "CONSIGHT An Adaptive Robot with Vision," Robotics Today, 1979, p. 26.

[16] J. Albus, E. Kent, M. Nashman, P. Mansbach, L. Palombo, M.O. Shneier, "Six Dimensional Vision System," SPIE, Vol. 336, Robot Vision, 1982, p. 142.

[17] R.C. Bolles, R.A. Cain, "Recognizing and Locating Partially Visible Objects: The Local Feature-Focus Method," Int. Journal of Robotics Research, Vol. 1, 1982, p. 57.

[18] R.C. Bolles, P. Horaud, M.J. Hannah,

"3DPO: Three Dimensional Parts Orientation System," Proc. of The Int. Joint Conf. on Artificial Intelligence, August 1983, p. 1116.

[19] J.A. Simpson, R.J. Hocken, J.S. Albus, "The Automated Manufacturing Research Facility of the National Bureau of Standards," Journal of Manufacturing System, Vol. 1, No. 1, 1983, p. 17.

[20] R. Lumia, "Representing Solids for a Real-Time Robot Sensory System," Proc. Prolamat 1985, Paris, June 1985.

[21] M.O. Shneier, E.W. Kent, P. Mansbach, "Representing Workspace and Model Knowledge for a Robot with Mobile Sensors," Proc. 7th Int. Conf. Pattern Recognition, 1984, p. 199.

[22] M.O. Shneier, R. Lumia, E.W. Kent, "Model Based Strategies for High-Level Robot Vision," CVGIP, Vol. 33, 1986, p. 293.

[23] A. Barr, E. Feigenbaum, The Handbook of Artificial Intelligence, (Los Altos, William Kaufman, 1981).

TABLE 1--OPERATOR INTERACTION AT EACH LEVEL

LEVEL	TYPE OF INTERACTION
At the servo	replica master, individual joint position, rate, or force controllers.
above level 1	joy stick to perform resolved motion force/rate control
above level 2	indicate safe motion pathways. Robot computes dynamically efficient movements
above level 3	graphically or symbolically define key poses. menus to choose elemental moves.
above level 4	specify tasks to be performed on objects.
above level 5	reassign telerobots to different service bays. insert, modify, and monitor plans describing servicing task sequences.
above level 6	reconfigure servicing mission priorities.

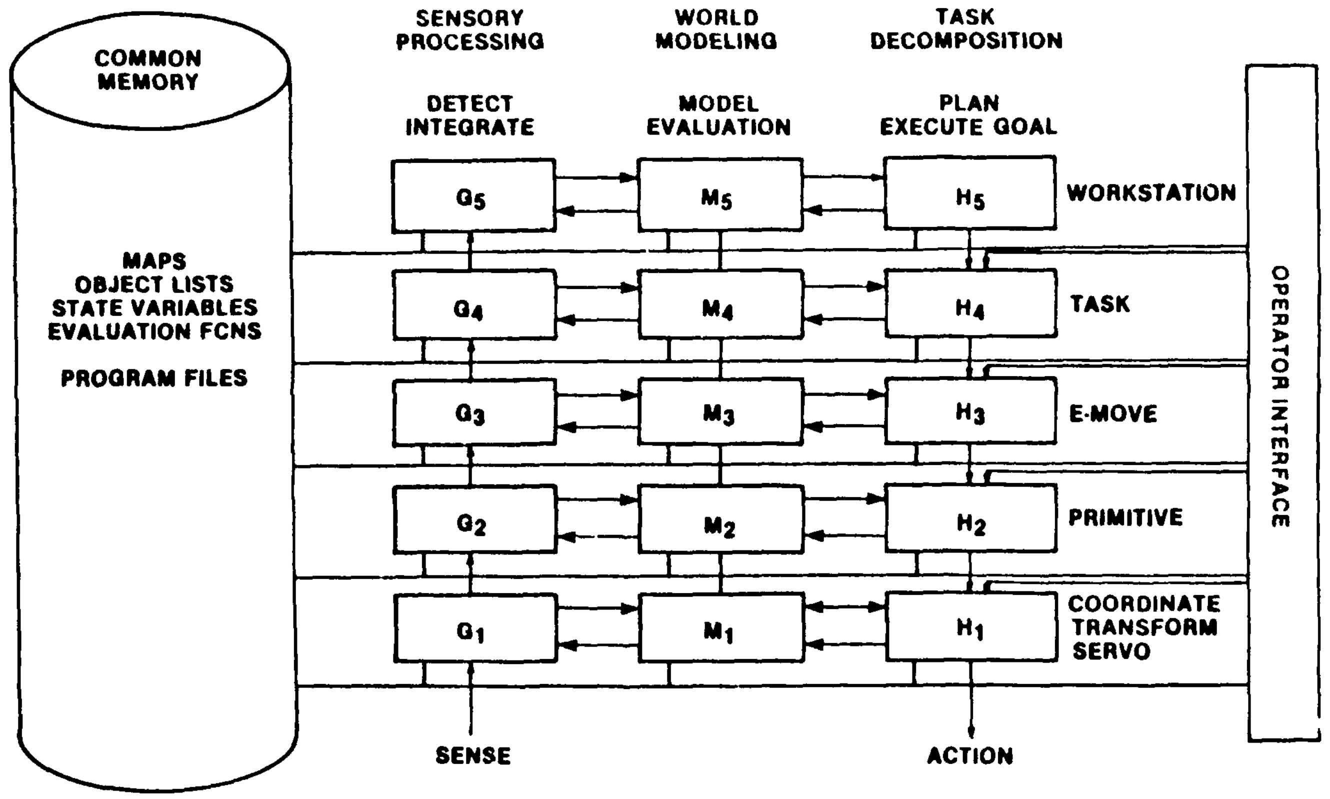


FIGURE 1 : A Hierarchial Control System Architecture for Intelligent Vehicles

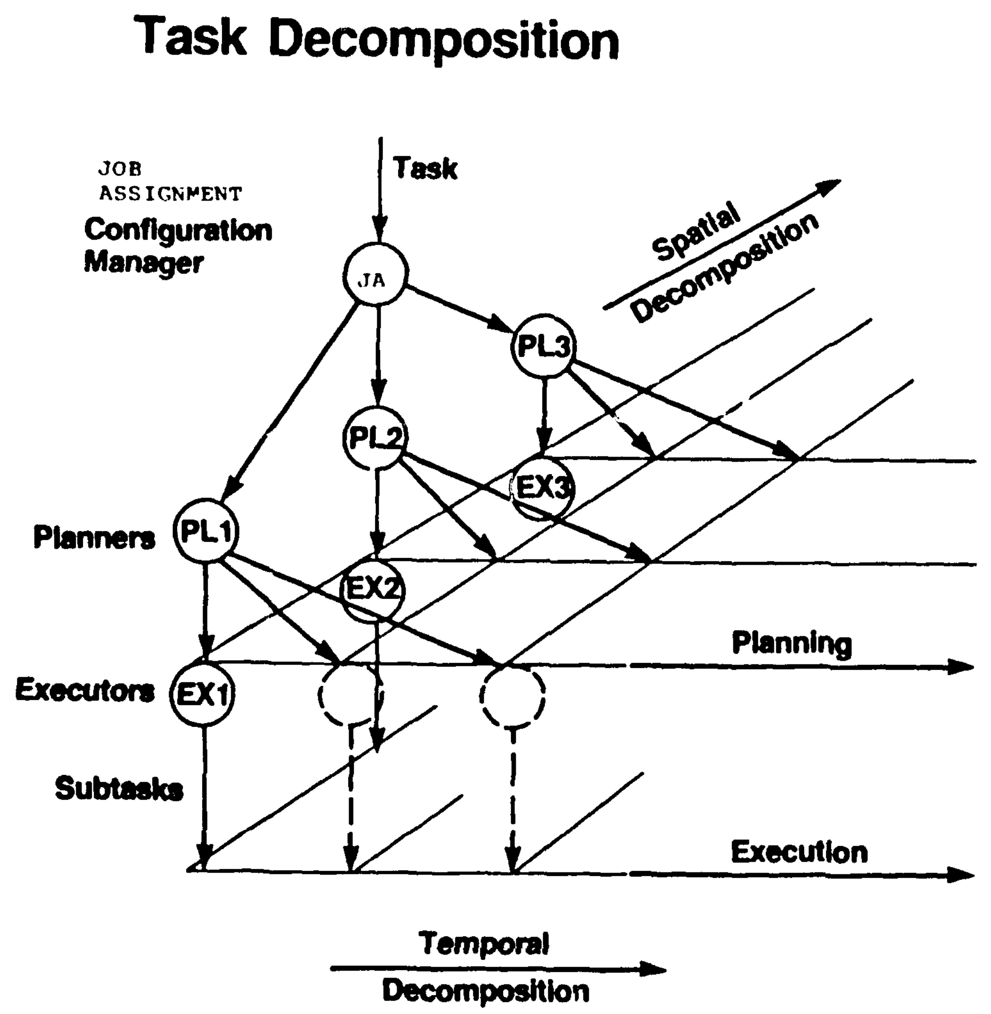


FIGURE 2: The job assignment JA performs a spatial decomposition of the task. The planners PL (j) and executors EX (j) perform a temporal decomposition

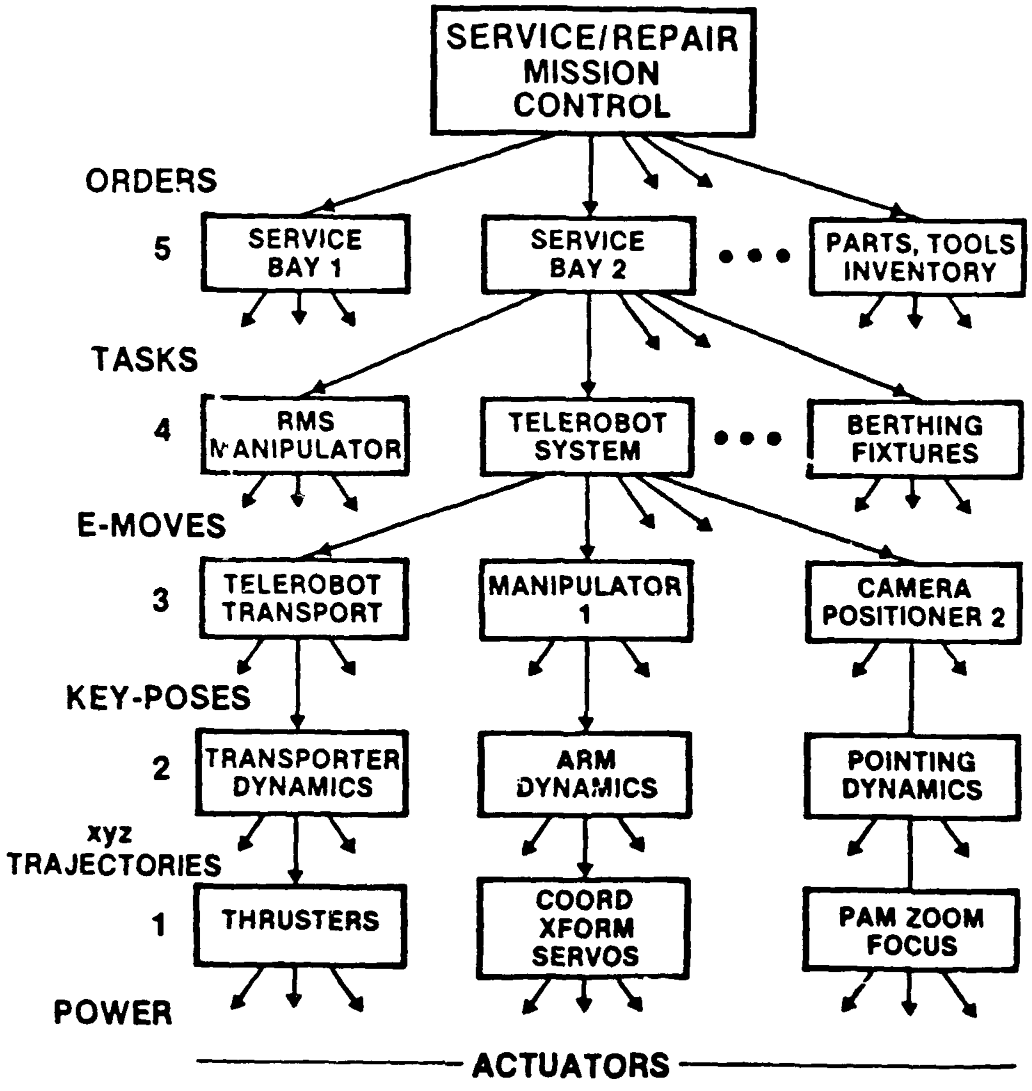


FIGURE 3: A six level Hierarchial Control System Proposed for Multiple Autonomous Vehicles

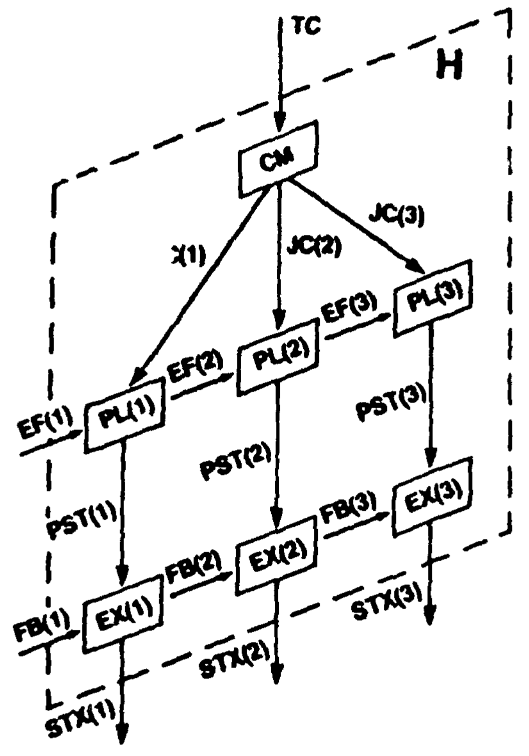


FIGURE 4: The H Module at each level has three parts: A job assignment module *JA*, *Planners PL* and a *set of executors EX*.

AFOSR-TR- 88 -0888

AIAA-87-1688
Integrated Army Robotics Thrust

C. M. Shoemaker, U. S. Army Human Engineering Laboratory, Aberdeen Proving Ground, MD

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

March 9-11, 1987/Arlington, VA

INTEGRATED ARMY ROBOTICS THRUST

C. M. Shoemaker
U. S. Army Human Engineering Laboratory
Aberdeen Proving Ground, Maryland

Abstract

The U. S. Army has recently defined and implemented a program to apply robotics technology for use in hostile environments. The program spans from teleoperated to highly autonomous applications of manipulator, vehicle and weapon technology. Each of the three application areas will be surveyed in this paper with an emphasis on the program concerned with robot manipulators.

Introduction

During the last two years, the U. S. Army has implemented a new program to develop the technology base required for robotic systems which can function effectively in the field environment. This effort is motivated by opportunities to reduce hazards to troops, demographic trends necessitating maximum productivity from available manpower, and the potential for reliable robotic systems to permit reductions in operation and support costs.

During late 1985, the Army Materiel Command Headquarters tasked the U. S. Army Human Engineering Laboratory (HEL), AMC's lead agency for field oriented robotics, to develop a thrust program in robotics which would achieve "critical mass" for a few key programs. The resulting program, for which the Army's Arroyo Center (Rand Corporation Federal Contract Research Center) provided analytical support, is a broad based effort involving a diverse array of research and development activities from the Army's Materiel Command (AMC) and Training and Doctrine Command (TRADOC), the Defense Advanced Research Projects Agency (DARPA), National Bureau of Standards, National Labs such as Oak Ridge and Sandia, and a number of universities. The pace and scope of the robotics thrust requires utilization of support from the groups identified above and the contractor community. Current industry participation in the program includes some of the largest corporations in the country as well as small innovative businesses.

Many of the technical challenges for this program stem from a need to function in a relatively unstructured environment which is, in many cases, actively hostile, with a further complication. A major complication results from requirements for many of the systems of Army interest to possess significant autonomous land navigation capabilities. These challenges, coupled with the Army's intent to actually field robotic systems of demonstrated value, has led to an emphasis on systems which span teleoperation for near-term capabilities thru supervisory control, in which the system has a significant level of autonomy, and the soldier acts as a [illegible] manager, to truly robotic systems in which human intervention in the system's operation will be infrequent. It has also led to a program investment strategy which includes one group of application areas, materials handling, in which considerable task structure is present. The materials handling applications will receive particular emphasis in this paper.

Figure 1 highlights the three major efforts comprising the Integrated Robotics Thrust. On the left are the wide range of application interests for the technology. (In fact, this is a small but representative subset of the application community's interests in robotics.) Next, the applications have been parsed into teleoperator and robotic systems. Although teleoperated systems do not meet the strict definition of robots due to their lack of autonomous capabilities, they play an important part in the program as near-term, jumping-off points for truly robotic systems. Next, the key technical areas associated with the use of robotics in the field environment are listed. The three major components of the program are listed under the column headed thrusts. A brief description of each program follows:

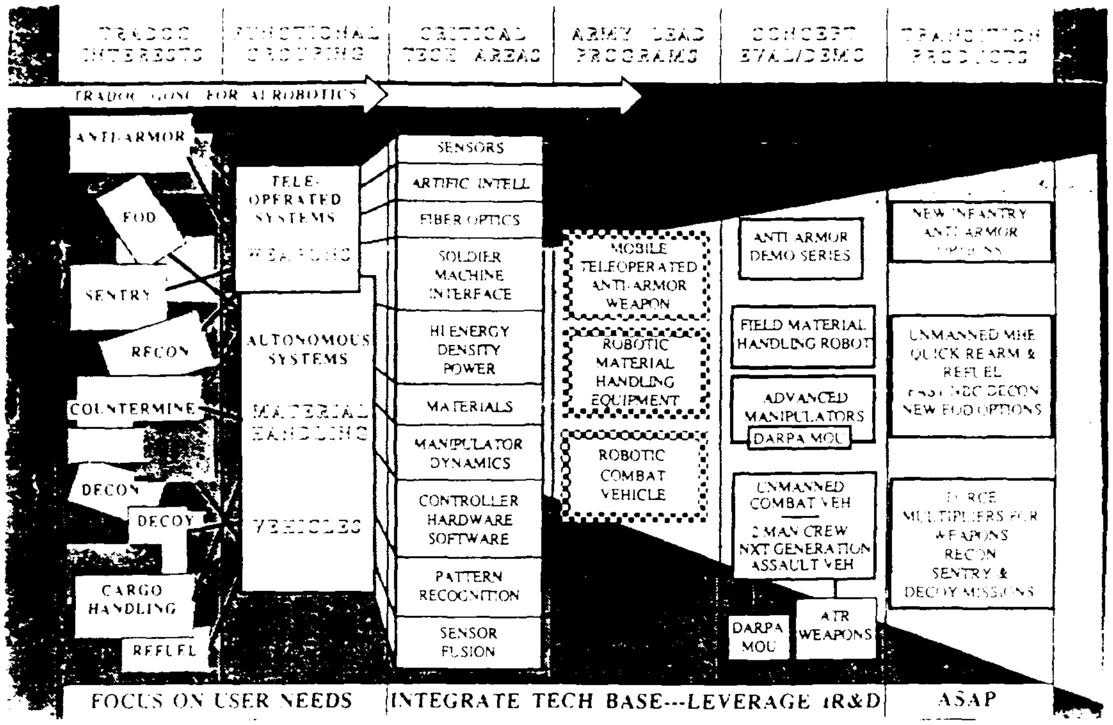


Fig. 1 Program Evolution

Teleoperated Mobile Anti-Armor Project (TMAP)

The TMAP project explores the near-term [illegible] opportunities that small, [illegible] -equipped systems will provide the infantry [illegible]. In concept, TMAP has a total system [illegible]. Major TMAP subsystems include a vehicle in the all terrain class, a [illegible] communication, a [illegible] control, and a [illegible] display and control input device [illegible] the system operator to deploy and operate TMAP from a remote location. The system operator will deploy the system, acquire, identify and engage targets using the TMAP sensor/weapons elements. These operations will be possible when the vehicular system is out of the operator's direct view. Major challenges of this program include the design and fabrication of an expendable system, [illegible] and sensor characteristics and the operator interface. The Army's Missile Command (MICOM) is the lead agent for this program, and the Infantry Center and School is the principal Training and Doctrine Command proponent.

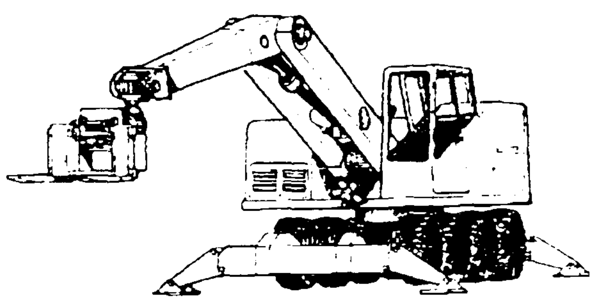


Fig. 1 Field Material Handling Robot

The performance of this system is dependent upon utilization of an advanced computer [illegible] package developed by the National Bureau of Standards, which includes the Real-Time Control System (RCS) and acoustic and proximity sensors located on the end effector for pallet [illegible]. Major challenges associated with the FMR include control of a robot manipulator with unprecedented levels of power, reach and speed in the field

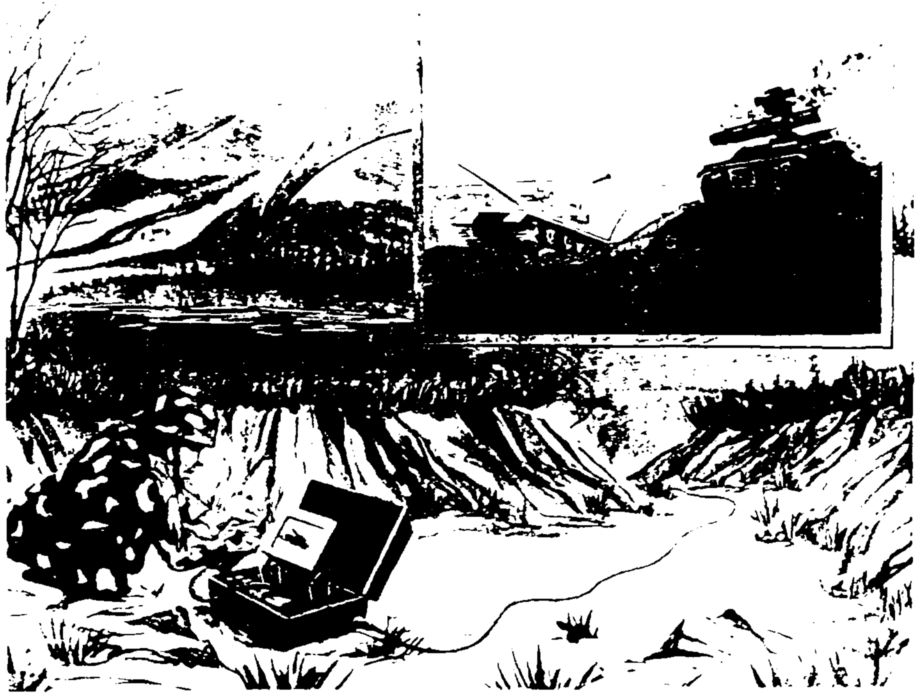


Fig. 2 Teleoperated Mobile Anti-Armor Project (TMAP)

Material Handling Robotics

This program is focussed on robotic manipulator systems. One of the two program sub-elements, the Field Material Handling Robot (FMR), will operate in a logistics workcell autonomously moving a wide variety of palletized loads weighing up to 2 tons, with a reach of 27 feet and a load handling cycle time of 20 seconds.

environment, and packaging of the system within C-141 aircraft weight and volume transportability constraints. Following two years of preparatory work by the Human Engineering Laboratory, the National Bureau of Standards and Belvoir Research and Development Center, a contract was awarded to Martin Marietta Aerospace Corporation in May of 1986 for development of a full-scale FMR demonstrator system. Although not elements of the at present, two other areas of future MHE robotics include vehicle systems which may ultimately interface with the Field Material Handling Robot. The first of these is the Palletized Loading System (PLS) shown in Figure 4.

Handling Robot in that precision [illegible] performance drivers. Although this system is primarily intended as a research [illegible] factors investigations in support of [illegible] such as explosive ordnance disposal, [illegible]

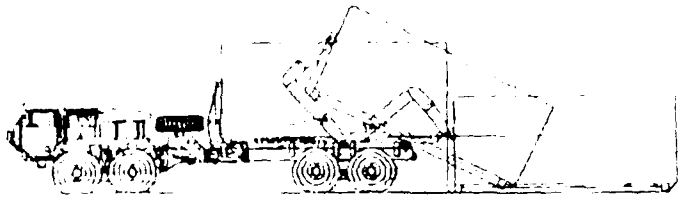


Fig. 4 Palletized Loading System

The PLS is a transportation asset which presents the opportunity for automated configuring of flat racks of mixed materiel for the field user. Rather than a flat rack consisting entirely of artillery projectiles, the situation may require a more self contained load of projectiles, propellant charges, and fuzes sent directly to firing units. An illustrative plan view of such a workcell is shown in Figure 5.

refueling of combat vehicles, decontamination and battle damage assessment, key elements of the system, e.g., manipulator, wrist and end effectors (hands) are being configured to ensure maximum technology transfer into appropriate mission areas. A key technical achievement related to SRIP has been the demonstration of an active manipulator deflection compensation system permitting significantly higher manipulator payload to weight ratio

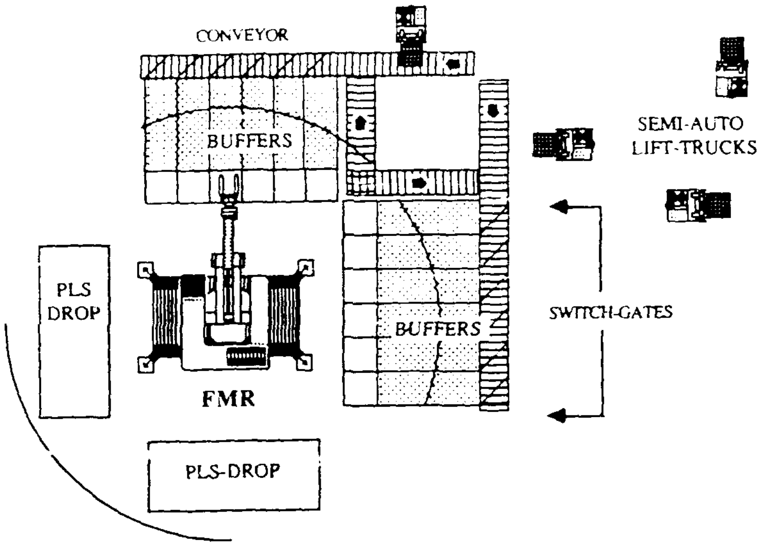


Fig. 5 PLS Automated Workcell

An alternative Field Material Handling Robot use is in conjunction with a highly automated ammunition resupply vehicle for the upload, transport and offload of ammunition for the artillery user. The illustration captioned artillery ammunition flow depicts a conceptual application of robotics to the ammunition supply system.

A second element of the material handling robotics program is the Soldier Robot Interface Project (SRIP). See Figure 7.

This program is developing a 7-foot reach, 150-lb robot manipulator mounted on a vehicle connected to a remote operator via fiber optics. The SRIP emphasizes the other end of the manipulator capability spectrum from the Field Material

than had previously been the case. SRIP is an example of direct synergy between the nuclear community and Army interests in field applications for robotics. The Army selected Oak Ridge National Laboratory as the system integrator for the SRIP project in recognition of thier vast teleoperation experience base in the nuclear community. The Army Laboratory Command is the developer for both of the material handling robot systems.

The Army will significantly benefit from the DARPA Advanced Manipulator Systems (AMS) program which will lead to improvements in the design of manipulators for field applications. Specific improvements are expected in force control, more dextrous multi-functional grippers and overall improvements in system performance, packaging and

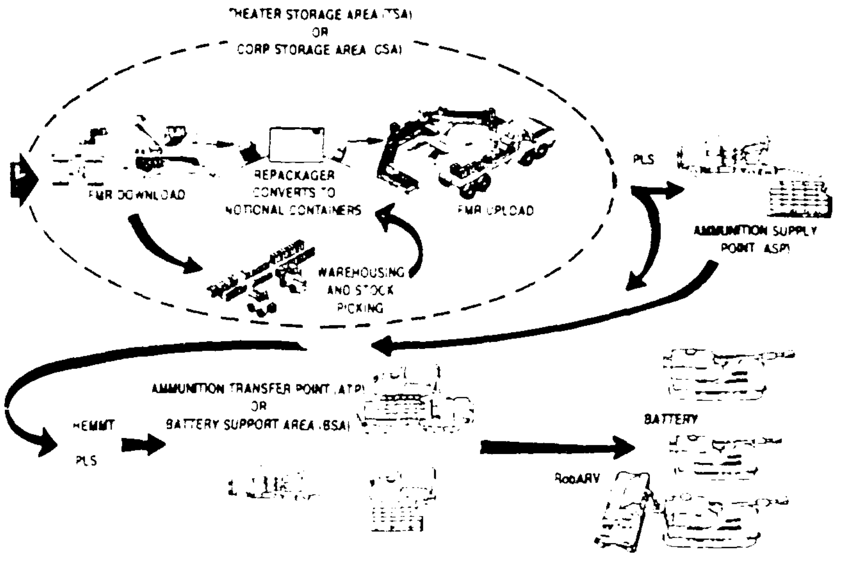

THEATER STORAGE AREA (TSA)
OR
CORP STORAGE AREA (CSA)
FMR DOWNLOAD
REPACKAGER CONVERTS TO NOTIONAL CONTAINERS
FMR UPLOAD
WAREHOUSING AND STOCK PICKING
PLS
AMMUNITION SUPPLY POINT (ASP)
AMMUNITION TRANSFER POINT (ATP)
OR
BATTERY SUPPORT AREA (BSA)
HEMMT
PLS
BATTERY
RobARV

Fig. 6 Artillery Ammunition Flow

[illegible] Human Engineering Lab in DARPA's [illegible] program.

Robotic Combat Vehicles (RCV)

The third element of the robotics thrust will apply robotics in the context of manned vehicles through the introduction of robotic autoloaders and target recognition systems into manned tanks. Beyond this application, force multiplication opportunities (and conversely command and control complexities) offered by multiple supervisory controlled unmanned robotic vehicles will be explored. The initial emphasis of RCV's unmanned [illegible] program will utilize the human as a supervisor [illegible] significant amounts of onboard autonomy to exploit opportunities for onboard [illegible] processing to permit the use of lower data rate communication links. Important sub-elements of the RCV program include: the new Laboratory Command cooperative laboratory initiative exploring the use of artificial intelligence capabilities for sensor data processing and command and control functions; the Tank Automotive Command's Advanced Ground Vehicle Technology Program (AGVT) for which FMC and General Dynamics are key contributors; and the Remote Command Center program will bring advanced control and display robotics technology to demonstrations of mobility oriented missions such as route reconnaissance.

Tank Automotive Command, Armament Research and Development Command, and Laboratory Command have

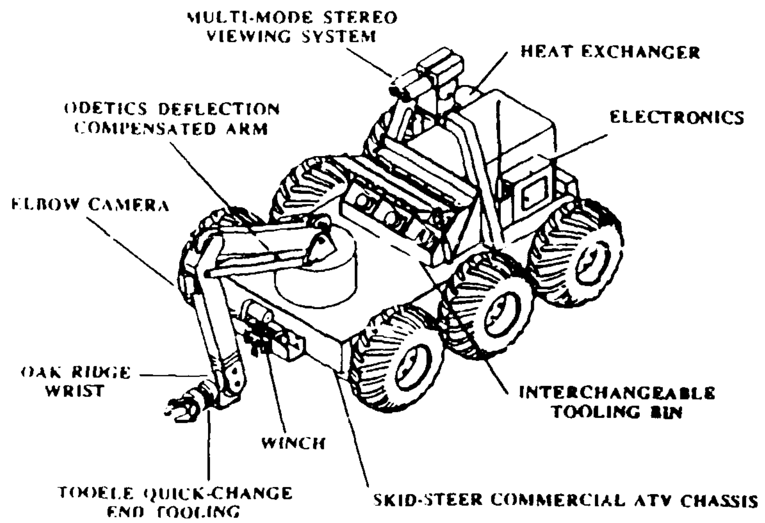

MULTI-MODE STEREO VIEWING SYSTEM
HEAT EXCHANGER
ODETICS DEFLECTION COMPENSATED ARM
ELECTRONICS
ELBOW CAMERA
OAK RIDGE WRIST
INTERCHANGEABLE TOOLING BIN
WINCH
TOOELE QUICK-CHANGE END TOOLING
SKID-STEER COMMERCIAL ATV CHASSIS

Fig. 7 Soldier Robot Interface Project

[illegible] in [illegible]. The Armor Center and [illegible] will be a principal component. DARPA is making a major contribution to this effort through its Strategic Computing program for which the Army's Engineer Topographic Labs is agent. One of the lessons learned from industrial robotics experience [illegible] the introduction of [illegible] in from the [illegible]. [illegible] implications [illegible], the [illegible] Army [illegible] a forum for [illegible] imple-mentation elements within the Army [illegible] focus exclusively on robotic technology. It must address the major challenges [illegible], command and control, training, and system reliability and maintenance which advanced automation brings with it.

The Army's commitment to explore the multi-faceted challenges of military robotics is [illegible] the activities of an extremely diverse array of agencies and people on these issues. This is a prerequisite for [illegible].

AIAA-87-1694

Design and Control of Modular Kinematically-Redundant Manipulators

J. P. Karlen, J. M. Thompson Jr. and
J. D. Farrell, Robotics Research Corp.,
Milford, OH

Second AIAA/NASA/USAF Symposium on
Automation, Robotics and Advanced Computing
for the National Space Program
March 9-11, 1987/Arlington, VA

DESIGN AND CONTROL OF MODULAR, KINEMATICALLY-REDUNDANT MANIPULATORS

James P. Karlen
Jack M. Thompson, Jr
James D. Farrell
ROBOTICS RESEARCH CORPORATION
2000B Ford Circle
Milford, Ohio 45150

Abstract

Kinematically-redundant articulated manipulators offer significant advantages in tool-handling dexterity and utility. This paper discusses a 7 DOF manipulator system in production for terrestrial factory automation applications and the design changes proposed to make the system appropriate for Space Station use. This manipulator system features modular construction, self-contained joint-mounted electric drives, high repeatability, and the potential for extraordinary dead-reckoning accuracy. Manipulators of varying size, capacity and redundancy may be built from a family of modules having output torques from 40 in-lbs to 40,000 in-lbs. The control has a heirarchical architecture designed for real-time, sensor-driven operation and employs an efficient proprietary algorithm which performs the transformation of 6 DOF cartesian space commands into 7+ DOF jointspace. This algorithm utilizes one or more criteria to resolve the redundancy, distributing motion gracefully to deal with singularities and controlling the pose of the manipulator in the workspace. The control algorithm runs effectively on single board microprocessor hardware. This manipulator system is seen by the authors to be a close fit to space telerobotics specifications, with existing models providing ready means for terrestrial testing and demonstration.

Introduction

Mechanical arms will be among the key functional elements in a variety of different robotic systems employed in future space operations by NASA, the USAF and industry. In the 1990s, robotic devices with arms will assist in the assembly, inspection, and maintenance of the Space Station, and in the servicing of satellites in geosynchronous orbit. Early in the next century, more sophisticated, elaborate and capable versions of these devices will play important roles in the exploration and commercial development of the Moon and solar system. Robots will serve as new tools for the Astronauts, augmenting their capabilities and making their jobs easier and safer. The next phases of space exploration and development will introduce human beings in large numbers to the space environment, leading to the establishment of permanent space communities. It is the authors' conviction that the appropriate application of robotics technology will accelerate that process.

Manipulator Arm Design Issues

A robotic manipulator is a powered mechanical linkage operating under interactive computer program control, used to position, orient, and apply forces and torques to a tool mounted to the distal end of the linkage. A number of different designs for such servomechanisms have been developed, principally for the automation of industrial processes related to automobile manufacturing, but also for remote manipulation tasks undersea and in hazardous radiation environments. The NASA/Spar Aerospace™ arm, or Remote Manipulator System (RMS), is the only space qualified manipulator extant. RMS has clearly demonstrated the utility of such machines in the new environment.

Established mechanical arm designs differ from one another in the number and type of joints incorporated and in their linkage geometries, in the placement and type of joint actuators, and in control architecture employed.[1] Industrial robot arms available today may have as few as three driven joints, or as many as seven. In principle, the more joints in the arm linkage, the more general purpose and versatile the manipulator. An arm must incorporate at least six degrees of freedom to provide complete control of position and orientation of the tool. It is worth reviewing the evolution of current mechanical arm designs when considering the application of established art to space automation.

Arm Geometries, Actuator Arrangements and Motion Controllers for General Purpose Manipulators

Six or more degrees of freedom can be disposed in an arm in several ways. Essentially the differences relate to the geometry of the mechanism used to translate a three degree of freedom wrist, or tool orienter, and the type of motion control system thereby required to produce straight line moves and other controlled paths at the toolpoint.

One basic form of manipulator employs a set of three slides connected at prismatic or sliding joints disposed in a nominally orthogonal arrangement to position the wrist. Such a device is typified by the IBM RS 1™ industrial robot. This "cartesian" geometry has a number of distinct advantages over other types, most

IBM is a registered trademark of International Business Machines Corp
Spar Aerospace is a registered trademark of Spar Aerospace Limited

notably that no coordinate transformations are required to generate useful motions at the toolpoint. Linear and circular interpolation are sufficient.

A second type provides a pair of slides connected at a prismatic joint, rotated on a base revolute or pivot joint to translate the 3 DOF wrist. The Prab Model FA™ is an example. The workspace of this type of manipulator is roughly cylindrical.

In a third common arrangement, the wrist is positioned in the workspace by a slide connected at a single prismatic joint to a pair of orthogonally mounted revolute joints. This arrangement was established and is typified by the Unimation Unimate 1000™ industrial robot. Theoretically, such a "polar" coordinate geometry produces a spherical workspace. In practice, mechanical design constraints generally restrict the useful working envelope to a rather thin spherical shell less certain significant conic sections.

More sophisticated motion control systems are required for arms with cylindrical or polar geometries than for those having cartesian geometries, since coordinate transformations must be performed to generate straight line moves at the toolpoint. In addition, manipulators which employ one or more slides with prismatic joints in their linkage, as a class, exhibit certain significant performance limitations of particular importance in prospective space applications. The relatively large size and inertia of the slides result in relatively high motive power requirements for a manipulator of given payload and workspace. Furthermore, the positioning slides often interfere with objects in the working area, when extended, and with objects behind the machine, when retracted.

Among the types of manipulators which incorporate slides, the cartesian systems tend to be the least spatially efficient, since the workspace is often completely surrounded by the framework of slides and supporting structure. To minimize spatial inefficiency and interference problems, a few polar type arms have been designed in which the slide collapses on itself as it is retracted. In the U.S. Robot Maker™, for example, the slide is composed of a set of shorter slides which telescope. In the NASA/Martin Marietta™ Viking Lander arm, a thin-wall steel tube that forms the slide when extended is made to collapse flat in cross-section by rollers as it is retracted, where it can be coiled compactly on a drum. Mechanical implementations of these designs tend to have comparatively poor static and dynamic performance characteristics, however, due either to the number of additional prismatic joints incorporated or to the very thin-wall cross-section utilized.

To improve dynamic performance and solve workspace interference problems, a linkage geometry which permits considerably more efficient mechanical designs has been devised and widely adopted. In this geometry, a series of relatively rigid link segments connected at revolute joints is used to position the wrist. This arrangement is commonly known as a "jointed arm" or "articulated" manipulator. They are also occasionally referred to as "anthropomorphic" manipulators, although, in fact, six degree of freedom jointed arms are kinematically more like "backhoes" than human arms, since the linkage operates in a fixed plane. With six degrees of freedom in the jointed arm, there is one or at most two distinct arm configurations for any given tool location and attitude. The advantages to be derived from incorporating extra joints in an articulated arm, "kinematic redundancy", will be discussed in subsequent paragraphs.

Theoretically, a jointed arm design of six or more degrees of freedom produces a roughly spherical working envelope. The principal advantages of a jointed arm manipulator, of particular merit in general-purpose arms for space operations, relate to the fact that when the arm linkage which positions the wrist is retracted, it folds up on itself, permitting the arm to be quite compact for a given working envelope and light in weight for a given payload.

Two distinct mechanical embodiments of the jointed arm geometry have gained acceptance in industry. In one, the actuators which drive the arm and wrist joints are mounted near the base of the machine, at some distance from the joints which they operate. In such designs, motors, gear reducers and position feedback devices located at the "shoulder" transmit power to the joints through the effects of a four-bar linkage, or through bell-cranks and pushrods, or by chains, timing belts or some other "tendon" arrangement. One excellent embodiment of this approach to a jointed arm is the ASEA IRb 6™ industrial robot. The remote-mounted drive approach has certain advantages, particularly in terrestrial robot applications where the requirements of the gravity field can be a predominant design factor. These advantages relate to the disposition of the mass of the joint drives off of the arm itself, and the consequent reduction of power requirements. Nevertheless, the drive train which is employed to transmit power to a remote joint itself imposes a number of significant performance penalties. It often restricts the range of motion of the joints, resulting in relatively limited working envelopes, torroidal in shape. A long drivetrain also adds considerable inertia, compliance and mechanical inaccuracy between the servo actuator and the joint, producing less than optimal dynamic response and stability.

In a second embodiment of the jointed arm geometry, substantially all of the joint drive actuators are mounted in or directly adjacent to the joints which they operate. A well-known example of a design in which actuators are mounted in the link adjecent to the driven joint is the Unimation PUMA™. This drive arrangement overcomes linkage travel problems, increasing the useful working envelope of the arm to the extent that, in some models, envelopes approach the theoretical sphere. Additionally, true joint-mounted direct-drive designs completely eliminate the drivetrain otherwise

Prab is a registered trademark of Prab Robots, Inc.
U.S. Robot and Maker are registered trademarks of United States Robots
Martin Marietta is a registered trademark of Martin Marietta Corp.
ASEA is a registered trademark of Asea Robotics, Inc.

required to transmit power to the joints from remote actuators, significantly reducing driveline compliance and inertia problems. In terrestrial robot applications, the significant penalty associated with mounting the drives in the joints relates to the additional weight carried in the arm itself. While the overall weight of most joint-mounted drive arms is, in fact, lower than that of most remote-drive systems of comparable dimension and capacity, the location of that mass is of great concern in a gravity field. Terrestrial joint-mounted drive arms which are not somehow counterbalanced may utilize as much as 80% of the actuator continuous torque ratings just to support the arm in its worst-case posture. The remainder is available to support and accelerate the payload. For space applications, the overall weight of the device is a critical issue, but the way in which the mass is disposed is of less concern.

From the viewpoints of spatial efficiency, tool-handling dexterity, size, weight and dynamic performance, an articulated arm geometry which utilizes joint-mounted actuators provides superior performance as compared to alternate configurations. The authors believe that such designs should be the leading candidates for space operations.

Jointed arm manipulators require relatively sophisticated motion control systems, capable of performing elaborate and time-consuming coordinate transformations to generate joint positions to locate and orient the tool. The complexity and cost of computer control systems required for accurate and, increasingly, adaptive control of the toolpath in jointed arm manipulators have had a significant influence on the particular linkage geometries utilized in most commercial versions of jointed arms. Kinematic geometries have generally been adopted which simplify the process of transformation from joint coordinates to cartesian space, and back, in an effort to reduce the number of and rate at which computations must be performed. For example, common jointed arm linkage geometries avoid rotated "off-set" pitch joints, a feature which has distinct advantages in increasing the useful workspace of the arm, but which greatly complicates coordinate transformation (Figure 1). By imposing specific constraints on linkage geometry, explicit mathematical expressions (i.e., closed form, analytic expressions) can be obtained for the coordinate transformations. In most commercial 6 DOF arms, the control system is simplified at the expense of manipulator dexterity and mechanism efficiency.

Kinematic Redundancy

Articulated manipulators which incorporate six joint-mounted drives, while more spatially and mechanically efficient than other general purpose arm designs, are considerably less maneuverable and dexterous than biological analogs they would ideally emulate, notably, the human arm or an octopus tentacle. As previously stated, six degree of freedom articulated arms function like backhoes from a kinematic viewpoint, in that the arm linkage extends and retracts in a fixed plane which

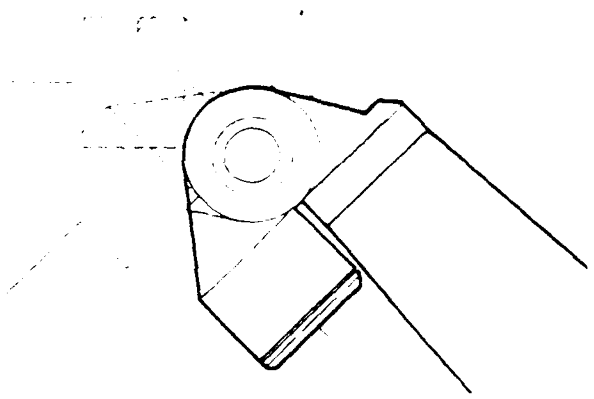
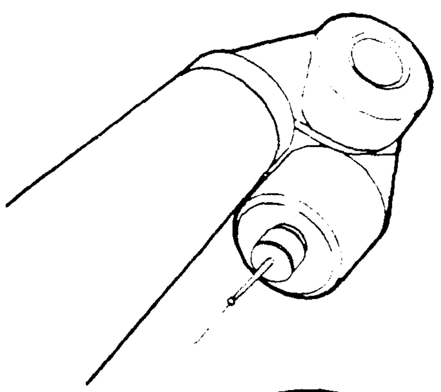
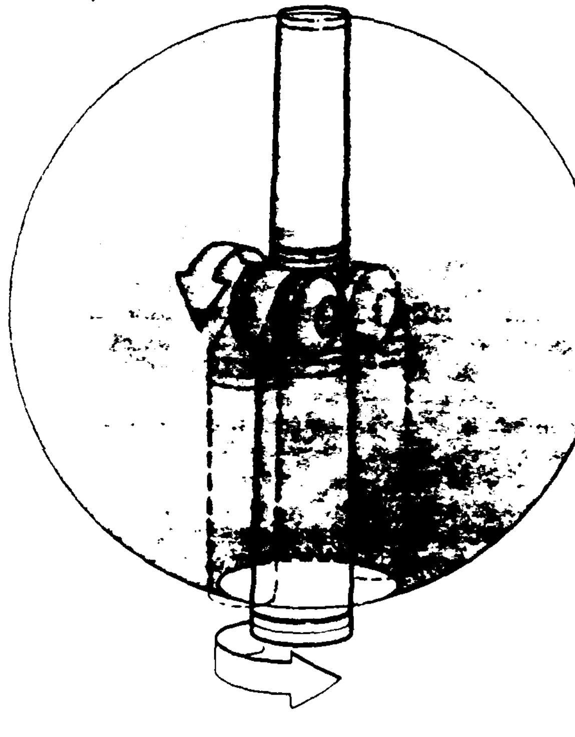




Figure 1. Off-set pitch joint kinematics.

s rotated about a vertical axis by the base revolute joint. With most such devices, a given location and orientation of the tool corresponds to a single discrete set of joint angles and an associated unique arm configuration. In a few such devices, a given tool location and attitude can be achieved by two discrete arm configurations. An example of a device with two possible arm configurations for a given tool position and orientation is the Unimation PUMA™. In that device, while the revolute joints representing the "shoulder" and "elbow" pitch axes remain fixed in a plane with respect to one another, the elbow joint can be disposed either "up" or "down". Nevertheless, if for a prescribed position of the tool, an obstacle in the workspace or the workpiece itself interferes with the arm link segments, the arm is not capable of reaching the goalpoint without collision (Figure 2). Unlike the human arm, a six axis jointed arm manipulator does not have sufficient degrees of freedom to reconfigure itself to reach around an arbitrarily placed object in the workspace. The human arm is considered by the authors to have seven degrees of freedom from shoulder to wrist, providing a range of possible elbow attitudes and resulting arm configurations for a given tool position and orientation.

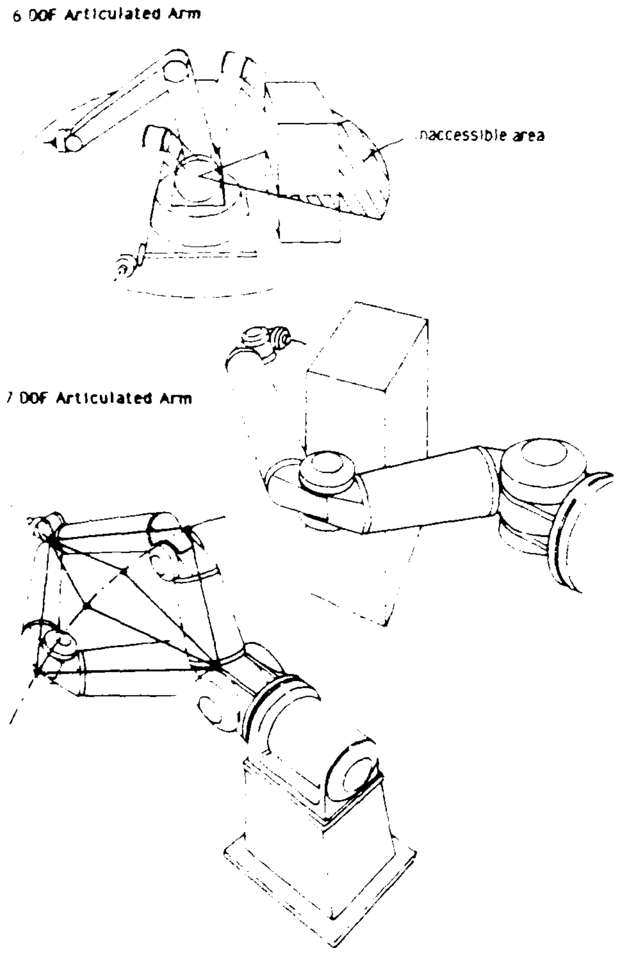


Figure 2. Maneuverability of a 6 DOF vs. 7 DOF articulated arm (Robotics Research Corp. K-2107 Dexterous Manipulator illustrated).

Many space servicing tasks will require the dexterity of a human arm. The authors believe that seven degrees of freedom should be considered the minimum incorporated in the tool-handling arms of a general purpose manipulation system, such as the NASA Flight Telerobotic Servicer (FTS). The authors term articulated mechanical arms having seven or more degrees of freedom in series "dexterous manipulators".

Additional degrees of freedom may prove extremely useful, as well. The addition of a three degree of freedom "torso", upon which is mounted two seven degree of freedom arms, may be an ideal general purpose manipulator system for prospective FTS assembly and servicing tasks.

Some tasks to be executed may require more than seven degrees of freedom in series in one articulated arm. The internal inspection of complex spacecraft assemblies with a hand-mounted camera, for example, could require the maneuverability of a snake. While such a task may seem unlikely at this point, and is certainly an unreasonable requirement for a baseline design for space manipulators, reconfigurable manipulator system designs which permit the ready addition of one or more joints (redundant joints) or the subtraction of unnecessary joints where the manipulator is dedicated to a single, constrained task) are seen by the authors to offer significant advantages over the long run.

The addition of one or more "redundant" joints in a manipulator has a number of significant benefits beyond flexibility or maneuverability.[2] In the same way that an extra degree of freedom provides means to reconfigure the arm to reach around an obstacle, the arm can be automatically reconfigured to dispose joints in a way which distributes torque or velocity requirements among the axes in an equitable manner. A human reconfigures his arm (and torso and legs) in the process of lifting a heavy object to keep the forces and moments applied at each and every joint appropriately distributed and minimized. A human uses the redundancy in his arm/torso/legs to shift his center of gravity and to maximize his "leverage". In a six degree of freedom jointed arm manipulator, no such reconfiguration of joints and redistribution of forces is possible. Thus, the mechanism's capacity to apply a particular force and torque vector at the tool may often be unreasonably limited because only a few joints are contributing to the exercise.[3] Coriolis forces, which could be employed to advantage, go entirely uncontrolled.

Equally important, intrinsic to all jointed arm manipulators is a condition known as a "singularity".[4] Singularities are regions within the working envelope of the linkage in which, either 1) no individual joint in the linkage lies in a position and an orientation in which it can contribute significantly to the commanded toolpoint direction within its physical limits of travel and acceleration; or 2) more than one joint in the linkage, independently, can execute the commanded motion.

The effect is the same. The arm linkage has lost one or more degrees of freedom. Arm behavior significantly degrades when the toolpoint is in the vicinity of a singularity, since the linkage is unable at that instant to maintain the commanded toolpoint position and/or orientation. The equations typically used to control motion at the toolpoint of a jointed arm have no mathematical solution at a singularity. While encountering one may be elementary, getting out can be difficult. Kinematic redundancy provides means to handle the problem of singularities. In a kinematically redundant manipulator, more than one set of joint angles can be employed to attain a commanded tool position. The authors (principally Farrell) and Håvard I. Vold have devised computationally efficient algorithms by which the axes in a kinematically redundant manipulator can be configured to contribute to any arbitrary commanded tool directional and rotational vector, enabling the arm to exit a singularity in a controlled fashion. This algorithm may have application in general purpose, kinematically redundant manipulator systems for space operations.

Special Purpose Manipulators

In addition to the four basic types of general purpose manipulator geometry reviewed above, each of which provides six degrees of freedom at the tool, many other linkage arrangements providing fewer controlled axes have been devised for special applications. In the design of most such special purpose arms, an effort is made to employ a linkage with the minimum number of driven joints to perform a unique tool trajectory or class of trajectories. Significant cost savings result from such an effort through the reduction in the number and often the size of structural components, joint actuators, power supplies and servo-control hardware.

Special purpose, limited degree of freedom manipulators have found wide use in industrial applications where a device is dedicated to one installation and task for its useful life. The application must be well defined and the associated trajectories must be fully constrained for these devices to perform the function. Nevertheless, the majority of factory automation applications are today being accomplished with manipulators which incorporate fewer than six degrees of freedom (five axes is the most prevalent geometry). Special purpose geometries are developed by careful selection of appropriate joint axes, link lengths and base location with respect to the workspace. Common examples of this specialization in industry include four axis lathe loading devices, and five axis arc welding robots.

Such specialization is natural, of course, and a similar trend can be anticipated in space robotics applications as time goes by. It may be determined, for example, that four degrees of freedom are fully sufficient to perform silicon furnace servicing operations in one bay of the Space Station. Possibly the correct three axis arm could perform many repetitive laboratory experiments. Some contractor studies suggest that the process of replacing ORUs on the bus of a satellite in geosynchronous orbit can readily be accomplished by a five axis manipulator carried by the Orbital Maneuvering Vehicle (OMV).

The authors believe that a wide range of different arm sizes and configurations will ultimately be determined to be appropriate for the various robotic and telerobotic space servicing operations. These may range in length from arms with the size and payload of the Remote Manipulator System, down to arms of less than human scale. They might ideally incorporate as few as three axes for certain applications, and as many as twenty for others. Each of these manipulators could be custom-designed for its task. The authors believe that a standard modular manipulator system which facilitates the assembly of a broad family of arms from a set of common joint-drive modules could provide a long term solution to this problem of specialization.

Motion Control Architecture

It is worth noting that the type of arm motion control computer system that is used to perform toolpoint trajectory control, coordinate transformation and servo-control, regardless of arm configuration, can be a relatively independent issue from what higher level control system is employed to generate arm commands. A properly segmented heirarchical control system, as suggested by James S. Albus, Anthony J. Barbera and Mary Lou Fitzgerald, decouples the arm motion control from higher level functions.[5, 6] In an *autonomous, sensory-interactive robot control system,* higher levels provide the user programming interface, accomplish task decomposition, break move statements into primitives and process information from global sensor systems, such as cameras. The arm motion controller accepts goalpoints in any arbitrary coordinate frame of reference and proceeds to execute the commands and report status. The motion controller concentrates on its local sensors, measuring joint position, velocity, forces and torques, internal temperature, and perhaps utilizes arm-mounted proximity sensors for local obstacle avoidance.[7] The higher level controller thus directs the arm as a peripheral. The "host" controller does not necessarily have to know what arm it is directing, or its unique geometry. This architecture is of particular significance in the design of robotic systems for prospective space operations for at least two reasons. First, with a heirarchical structure, any particular arm can be driven as a peripheral either by an autonomous robotic control system, or by some man-in-loop, teleoperated master controller. The arm does not recognize the difference. This has important implications for the feasibility of a standardized manipulator system for space operations. Secondly, the heirarchical control system might be viewed as a structured, "from the ground up" approach to implementing artificial intelligence. Each heirarchical level operates relatively independently as a closed-loop controller. In this context, a heirarchical robotic control architecture provides ready means for system expansion and enhancement as improved sensors and algorithms are devised for any given level

Conclusion- Opportunities for a Standard Modular Manipulator System for Space Operations (SMMS)

In conclusion, the authors believe that a considerable body of evidence indicates that the appropriate baseline configuration for the dexterous mechanical arm systems to be used for tool-handling in space servicing tasks, whether operating under "autonomous" computer program control or teleoperated, is an articulated, seven degree of freedom device having modular, joint-mounted drives and a standardized, local motion controller.

Furthermore, the authors propose that NASA and the USAF, in concert, consider developing and adopting a Standard Modular Manipulator System (SMMS) for all robotic and telerobotic assembly, retrieval, experiment support and servicing applications on the Space Station and for on-orbit remote satellite servicing.

SMMS is envisioned by the authors as a family of articulated manipulators ranging in size from less than three feet to over fifty feet in length, incorporating as many joints as necessary for a given application, each manipulator being an assembly of unitized joint drive modules of common design (Figure 3).

Lightweight, interchangeable joint drive modules, each having as few as two moving parts, could be designed to provide a set of standard revolute "roll" and "pitch" joints with output torque capacities ranging from 40 to 40,000 in.-lbs., in appropriate steps. Individual joint drive modules would carry across the family of arms. For example, a 2,500 in.-lb. pitch module which serves as the elbow in a 20 foot long manipulator would also be used as the wrist pitch module in a fifty foot arm and the shoulder pitch module in a five foot arm. In addition to their use as arms, these same modules could be assembled into configurations which serve as torsos for multiarm devices and, indeed, "legs" for certain vehicles operating on the Space Station truss (Figures 4 and 5).

The authors propose that an advanced motion controller conforming to a standard heirarchical architecture be utilized for each manipulator assembly. Each motion controller would have the capability to use an array of local kinesthetic sensors to perform the "brainstem" functions associated with the arm. In general purpose assemblies, incorporating seven or more degrees of freedom, the motion control functions would include coordinated joint control, real time, sensor-based arm obstacle avoidance, and optimization of arm posture for leverage.

With this architecture, SMMS arms could operate as intelligent peripherals to any task-appropriate host computer through a common hardware/software interface, facilitating use of a standard set of manipulation hardware in a wide variety of autonomous and teleoperated systems.

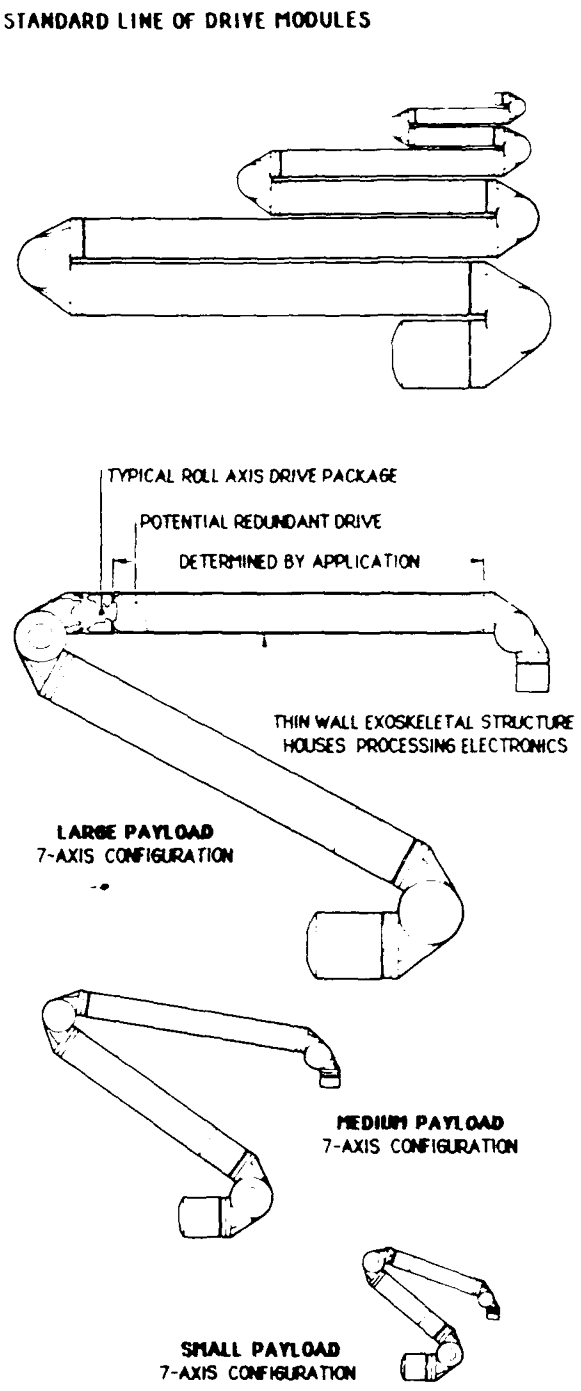

STANDARD LINE OF DRIVE MODULES
TYPICAL ROLL AXIS DRIVE PACKAGE
POTENTIAL REDUNDANT DRIVE
DETERMINED BY APPLICATION
THIN WALL EXOSKELETAL STRUCTURE HOUSES PROCESSING ELECTRONICS
LARGE PAYLOAD 7-AXIS CONFIGURATION
MEDIUM PAYLOAD 7-AXIS CONFIGURATION
SMALL PAYLOAD 7-AXIS CONFIGURATION

Figure 3. Robotics Research Corp. concept for a Standard Modular Manipulator System (SMMS) for space operations.

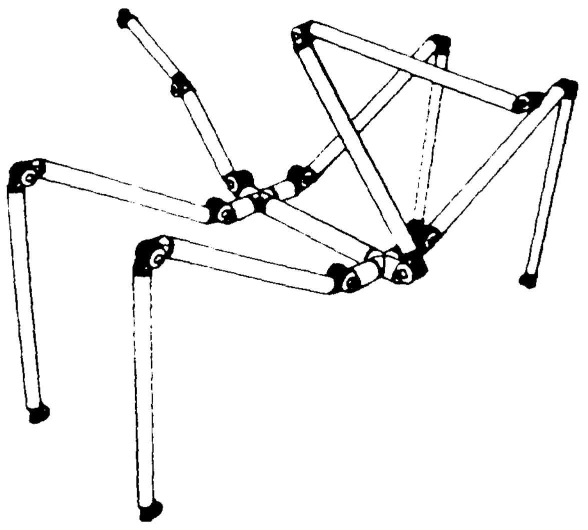


Figure 4. RAM/VATS concept for a legged vehicle for Space Station servicing employing SMMS modular manipulators.

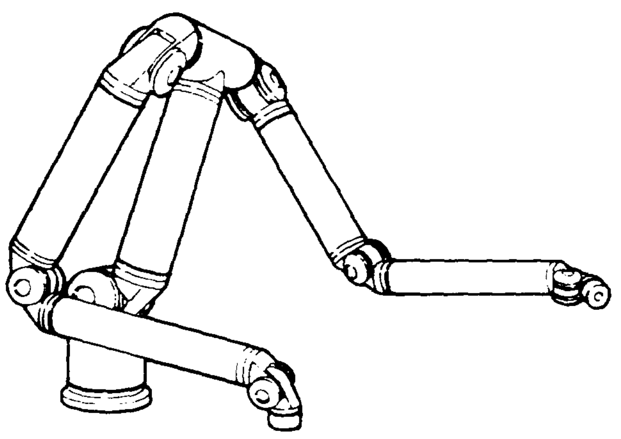


Figure 5. Robotics Research Corp. concept for a 17 DOF anthropomorphic manipulator for Space Station assembly and servicing (built from SMMS unitized joint drive modules).

The authors anticipate that the standardization of mechanical designs and control interfaces proposed in SMMS would substantially enhance manipulator system reliability, would minimize development and production costs, and would readily accommodate future growth in space operations and changes in mission.

A Prototype for SMMS

At Robotics Research Corporation, in Milford, Ohio, the authors and their colleagues have developed and demonstrated a modular, articulated manipulator system which embodies many of the characteristics required for SMMS. The "K-Series" Dexterous Manipulator system, while designed in scale and details for terrestrial factory automation and laboratory use, may serve as a reasonable prototype for future space manipulators. Current production models may provide ready means for terrestrial demonstration and experimentation with kinematically-redundant manipulators.

The K-Series Dexterous Manipulators are built up from a family of unitized joint drive modules. Each module includes a complete joint drive mechanism, feedback transducers, bearings and structure for a degree of freedom. In existing units, individual joint drive modules do not contain the signal conditioning, control and servo power electronics. These components are housed in a control cabinet and connected to the modules with a highly flexible internal wiring harness. Each module is designed around a particular size (and thereby torque capacity) harmonic drive reducer and incorporates a DC servomotor of appropriate power output. This drive system can be tailored to meet a wide range of application requirements by selecting an appropriate harmonic drive ratio between 60:1 and 320:1. The modules are of two types, configured as "roll" and "pitch"-type joints. Roll modules effect rotary motions about the axis of the module interface flanges, while pitch modules effect rotary motions about an axis perpendicular to the normal vector of the attachment flanges. Modules are joined to each other by V-Band clamps, secured by a single tangent bolt.

These roll and pitch-type modules are designed to be assembled, in series, into a variety of different arm sizes, payloads and kinematic configurations. Individual joint modules can be utilized across a family of arms (Figure 6). Two models in the line are shown in accompanying photographs. The K-2107HR is a seven foot long, seven axis manipulator configured specifically for applications which require a light-weight tool or sensor to be conveyed about a large working envelope with exceptional dexterity and speed, and with the greatest possible positioning repeatability (Figure 7). Operating in a stable temperature state, the K-2107HR will repeatedly position a tool to within 4/10,000ths of an inch. The K-1607 is a five foot long, seven axis unit configured for general-purpose factory and laboratory use (Figure 8).

Kinematically-redundant versions of K-Series manipulators (incorporating seven or more degrees of freedom) offer tool-handling dexterity which approaches that of a human arm. The seven degree of freedom arm configuration has, in fact, the same number and a similar disposition of joints as a human arm The K-Series manipulator actually eclipses anthropomorphic capabilities in range of motion for arm roll joints, producing +/-180 or +/-360 degrees, where

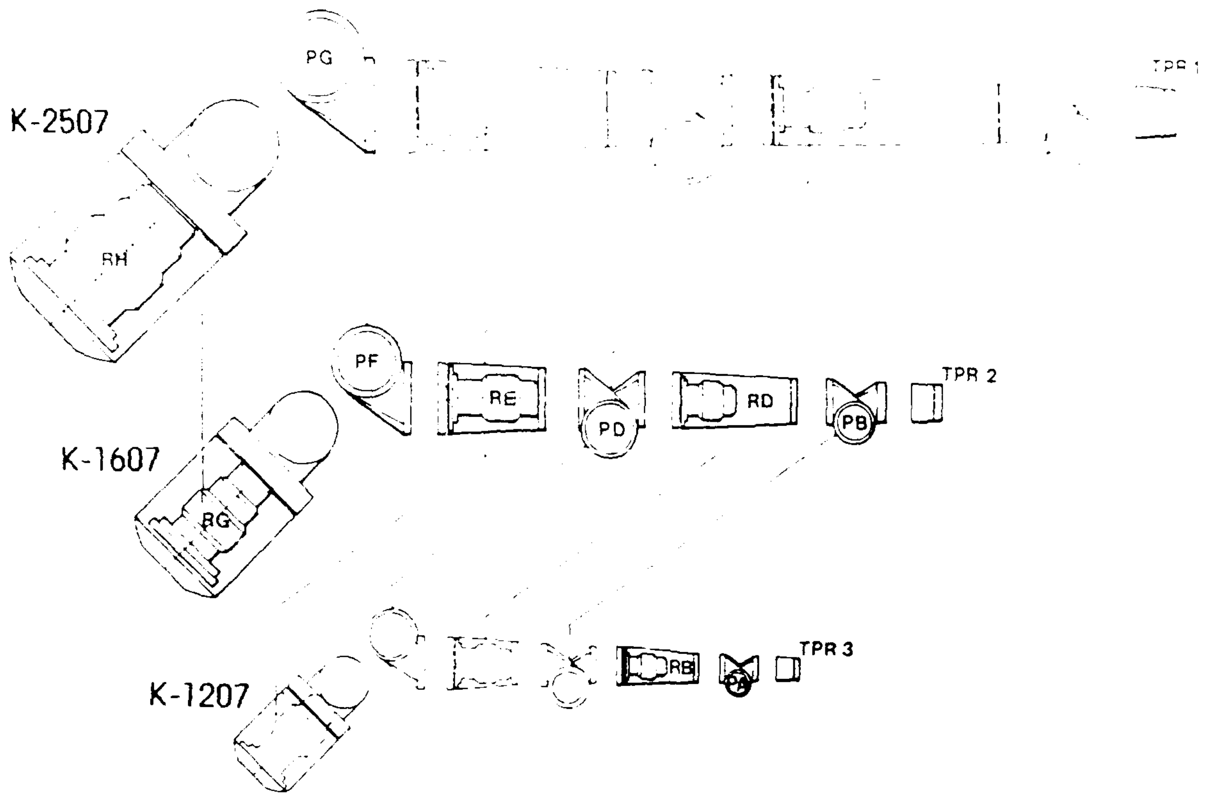


Figure 6. Robotics Research Corp. K-Series Family of Interchangeable, Unitized Joint Drive Modules.

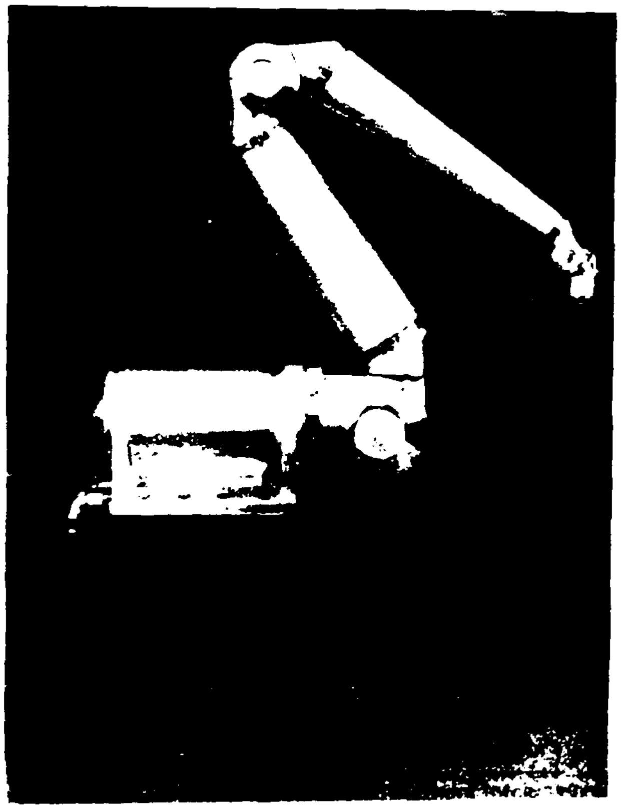


Figure 7. Robotics Research Corp. K-2107HR Dexterous Manipulator (7 DOF)

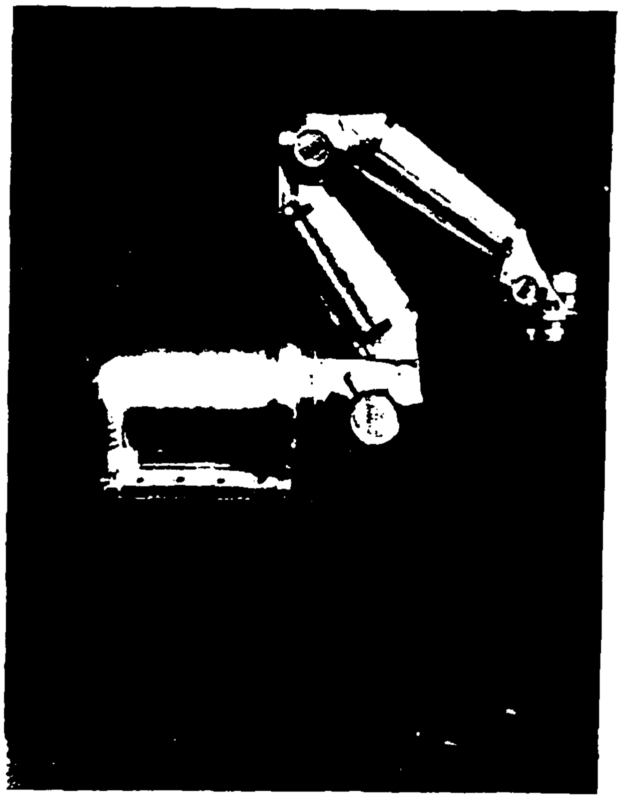


Figure 8. Robotics Research Corp. K-1607 Dexterous Manipulator (7 DOF)

the human arm joints are typically limited to +-45 degrees or less. The motion range available in the K-Series pitch modules slightly exceeds anthropomorphic, with a full 180 degree range. This range of travel in pitch joints is possible through the use of an offset geometry, where the pitch joint axis is displaced from the attached roll axes (refer to Figure 1).

The disposition of modules within a K-Series manipulator follows the pattern of roll-pitch-roll-pitch-etc. If a manipulator of reduced degree of freedom is desired, the most reasonable degeneration of capability is to replace a roll module position in the kinematic chain with a static link (i.e., a tubular structural element which has attachment provisions on both ends for pitch modules).

The generic actuator design utilized in all K-Series modules consists of a harmonic drive located on the joint axis, directly coupled to the proximal and distal castings of the module. A high performance samarium-cobalt DC servomotor is directly coupled to the harmonic drive wave generator. The flexspline of the harmonic drive couples to the structure through a metal-to-metal slip clutch and torque transducer. The slip clutch is generally adjusted to slip at a torque greater than required for full machine performance, but less than would damage the drive or other manipulator components. An independent, high precision instrument gear system drives a brushless resolver which is utilized to provide joint position and non-quanitized velocity feedback by means of an advanced R-to-D chip. By this arrangement, the servo system has applied actuator torque, joint velocity and joint position feedback available to provide controlled joint or axis motion.

The utilization of the harmonic drive reducer in this fashion is both effective and problematical. The harmonic drive is unrivaled for compact, light, backlash-free torque multiplication. It also exhibits a two-per-input-revolution transmission error which excites the inevitable resonance resulting from the inherent reducer compliance. This often-suggested arrangement of actuator system is typically handicapped by an obvious and objectionable resonance in operation. The resonance phenomenon intrinsic to direct-drive arrangements of harmonic drives in robot arms has prevented the widespread application of this otherwise attractive actuator package. K-Series manipulators utilize a servo-control approach which overcomes this problem and provides important attributes for advanced actuator control and improved manipulator performance. The conventional approach in robot arms is to control velocity and position of the motor shaft, while assuming that the transmission elements are nearly ideal in their translation of shaft motion into manipulator motion. The approach taken by on of the authors (Thompson) and colleague Paul H. Eismann in K-Series servo drives is to treat the motor and harmonic drive as a torque producer. The control feedback parameters are all measured at the interface between proximal and distal joint elements. The joint position commands joint velocity, which commands applied actuator torque. The innermost loop is thus a *torque loop* capable of bandwidth which encompasses the normal resonant frequency range of the actuator package. This torque loop can be visualized in several ways. One view is that it operates as an active damper for the joint drive resonance. Another view is that it provides the designer an opportunity, within the achievable loop bandwidth, to fundamentally re-write the constituent equation governing the behavior of the drive as a torsional spring between two inertias. The control loop actually functions to position the motor inertia, however it must, to cause the drive to provide the desired applied joint torque.

As previously mentioned, the torque-loop system provides significant advantages beyond eliminating the harmonic drive resonant response to provide smooth motion. A significant trend in research on manipulator control is to command axis torques to provide high-bandwidth tool force control, or "impedance control". In K-Series arms the fastest loop at work in the servo control system is the torque loop. This innermost loop can remain in operation while mode switching. (Joint torque transducers are intrinsic elements to each joint in the machine.) The torque loop also encompasses and compensates for motor, motor seal and drive friction. Viewed from outside the loop, it imparts to the system most of the attributes of best direct-drive manipulators, while avoiding the relatively poor torque-to-mass ratios of the direct-drive motors.

The K-Series module system utilizes an exoskeleton structural approach. The exoskeleton structure provides favorable structural dynamics of the overall manipulator, with low overall weight. It also provides a strong, durable and clean exterior, enclosing all wiring and actuator componentry.

A standard "black-box" motion controller has been developed to control any arm configuration that can be assembled form K-Series modules. This accepts position commands from any host, in global coordinates or some arbitrary frame of reference, and performs the trajectory control and coordinate transformations necessary to effect the commanded toolpoint motion. Such global goalpoints can be up-dated on a 50-100 millisecond basis, providing sensory-interactive control capability. The architecture of this motion control is illustrated in the accompanying figure (Figure 9).

The algorithm devised by the authors and Vold to accomplish coordinate transformations for 7+ degree of freedom manipulator configurations accomodates singularities in the workspace, provides means to employ the additional degrees of freedom to avoid obstacles, and to distribute in a reasonable manner the actuator torque requirements amongst the joints. The algorithm can be configured to operate any number of manipulator joints. The more degrees of freedom in the the manipulator, the more graceful the arm motion.

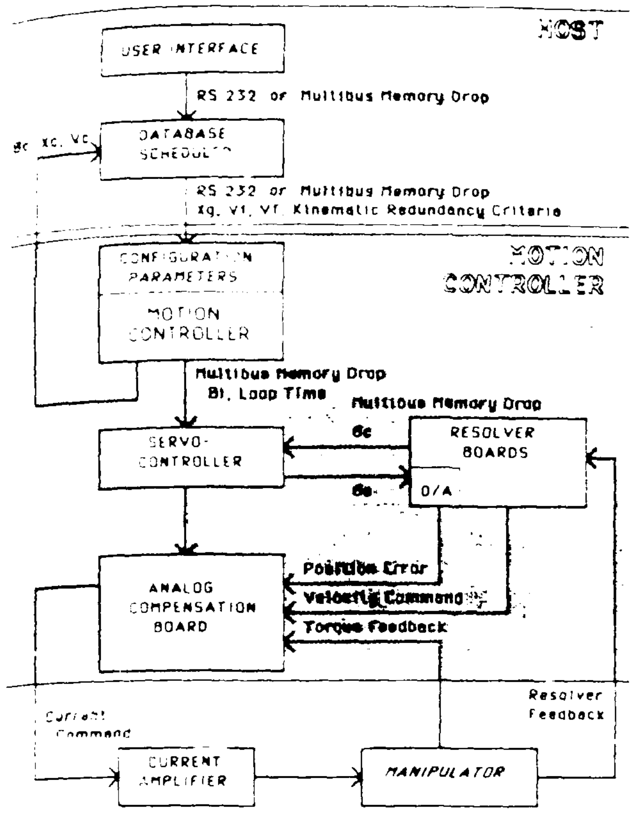


Figure 9 Robotics Research Corp. K-Series Motion Control System Architecture.

The K-series Dexterous Manipulators should provide appropriate test-beds for terrestrial research programs relating to future space manipulation systems, e.g., in the areas of adaptive force/position control, advanced teleoperation techniques and multi-arm coordination (Figure 10).

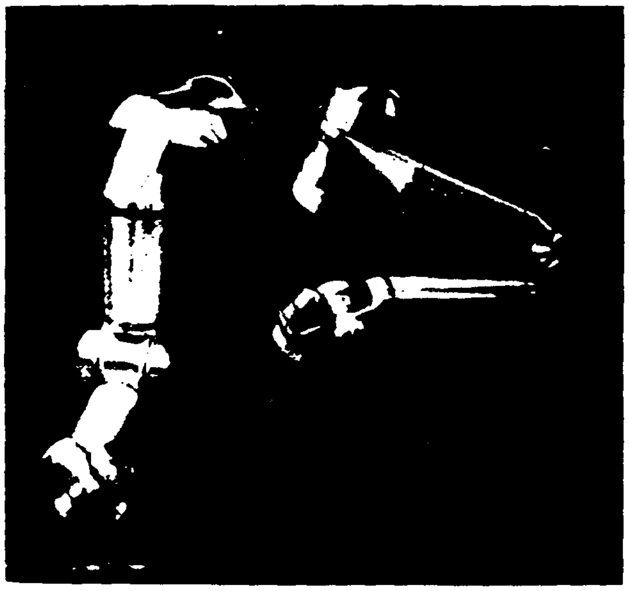


Figure 10 Dual-arm arrangement of K-1607 Dexterous Manipulators

Adapting the K-Series Design to Space Applications

The K-Series Manipulator System is seen by the authors to embody the fundamental mechanism and control technology required to implement dexterous tool-handling manipulators for space servicing applications, including the overall SMMS concept for a family of modular arms. The developments required for a space version appear relatively few and straightforward.

The high vacuum environment will necessitate the use of brushless motors. A brushless system offers significant improvement in arm thermal management and reduces the electromechanical system to an extremely simple, reliable package. Brushless motors will improve system reliability by providing redundant electronic systems. Their only disadvantage is that they require additional interface wiring between motor position transducers and the power electronics.

Harmonic drive lubrication needs to be carefully considered, with respect to outgassing, vacuum stability and general longevity. The harmonic drive has been used extensively in the space program, and there is no shortage of experience with this issue.

Thermal management of the manipulator needs to be addressed to assure acceptable power consumption and operational reliability. Possibilities exist to achieve thermal control by active configuration of insulation, reflectivity, and radiation of the manipulator itself. Other possibilities include utilizing heat pipe technology and insulation to control the thermal state of the manipulator as an extension of the thermal environment of the spacecraft itself. Both of these approaches could be augmented by phase-change thermal storage systems to extend the thermal time constant of individual actuator systems.

Individual joint drive modules need to be designed as Orbital Replacement Units (ORUs). In order to repair a manipulator which has malfunctioned by exchanging modules (possibly using another manipulator), everything which makes the module operate must be contained within the module, or ORU, itself. Taking the ORU concept and the overall system mass and reliability goals into consideration, one clear strategy for design becomes integrating the actuator, transducers, signal conditioning, analog and digital control electronics, brushless motor power and commutation electronics and a command/communication interface into one ORU. Each ORU might include one or two axes of motion, depending on replacement strategy and reliability analysis. Consolidation of these system elements into a ORU package will minimize the complexity of the flexible wiring harness and many of the potential failure modes associated with the harness. Instead, a power bus and a communication bus would be incorporated, each triple-redundant. Communications could be through optical fiber or utilize powerful bus communication techniques. All that would remain of the off-board control box is a power bus controller and motion control computer. Interim space manipulators can certainly be constructed short of this level of integration, but there are significant reliability advantages to be gained.

Preliminary designs of ORU modular manipulators currently underway at Robotics Research Corporation indicate very favorable characteristics may be achieved with respect to arm size, mass and reliability.

References

1. Roth, B., "Kinematic design for manipulation," *NSF Workshop on Research Needed to Advance the State-of-Knowledge in Robotics*, U. Rhode Island, April 15-17, 1980.

2. Yoshikawa, T., "Manipulability and redundancy control of robotic mechanisms," *Proc. IEEE Conf. Robotics and Automation*, St. Louis, March 25-28, 1985.

3. Hollerbach, J. M., and Suh, K. C., "Redundancy Resolution of Manipulators through Torque Optimization," MIT Artificial Intelligence Laboratory, Cambridge, MA, 1985.

4. Paul, R. P., *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, Cambridge, MA, 1981.

5. Albus, J. S., "Data Storage in the Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement and Control Trans. ASME*, Series G, Sept., 1975.

6. Barbera, A. J., "An Architecture for a Robot Heirarchical Control System," National Bureau of Standards, Dec., 1977.

7. Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal of Robotics Research*, Vol. 5, No. 1, Spring, 1986.

AFOSR-TR- 88-1508

AIAA-87-1695
NASA's Technology Plans — Will Technology Be Ready When We Are

R. S. Colladay, NASA Headquarters, Washington, DC

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

March 9-11, 1987/Arlington, VA

NASA'S TECHNOLOGY PLANS
WILL TECHNOLOGY BE READY WHEN WE ARE

Mr. Raymond S. Colladay*
Office of Aeronautics and Space Technology
NASA

ABSTRACT

NASA has embarked on a $178 million space research and technology program over five years to begin to restore the agency's technical strength. The program, called the Civil Space Technology Initiative, will contribute to the technology base that will help reestablish the U.S. civil space program to a position of world leadership. CSTI includes research in technologies to enable efficient, reliable access to and operations in space, and for technologies supporting science missions.

Transportation technologies include reusable, high-performance engines for next-generation vehicles, booster technology options, and an aerobraking flight experiment. Earth-orbiting operations include an expanded effort in automation and robotics and high capacity power. Science technologies include a large flexible structures and control program, sensor devices and high rate, high capacity data systems. CSTI is a means to many ends.

Historically, technology has been a cornerstone of our leadership in space and a key factor to U.S. industrial competitiveness. CSTI is a first step to maintaining that trend and will be implemented in close cooperation with industry and through expanded involvement of universities.

BACKGROUND

For twenty-eight years NASA has pushed at the cutting edge of space technology and at the frontiers of space. But now the Nation is at a decision point for a renewed dedication to the civil space program -- a new beginning. Why? Because the once-held U.S. position of dominance in space technology has been seriously challenged. Our direction as an agency, then, is to reconstitute the Nation's ability to supply space transportation services that will preclude interruptions in the future, and to strengthen the technology base with options that can sustain leadership in space. It will take world-class facilities, dedicated pursuit of science and advanced technologies, and, more importantly, developing and nurturing the human resources capable of meeting the challenges and adventures of our future endeavors. This challenge requires both our energy and money. It requires new technologies that merge the safety lessons expounded by the Presidential Commission on the Challenger Accident and the goals of expanded future space activity set forth by the National Commission on Space.

DECLINING INVESTMENTS AND COMPETITION

The National Commission on Space, in recognition of the need to substantially increase the national investment in space technology, recommended "A threefold growth in NASA's base technology budget...." This "three times" budget should be met in five years, for, no matter how clear a vision we have of our long range needs, we cannot move forward agressively in all necessary disciplines without a reversal of the budget trend. Historically, technology has been a cornerstone of our leadership in space and a key factor to U.S. industrial competitiveness. After peak investment in the early 1960's, the NASA Space Research and Technology budget fell precipitously so that within ten years we dropped to a relatively constant investment of only one fifth to one sixth of what it had been when the nation made its commitment to leadership in space. Our aerospace industry, the only remaining high-technology segment of the U.S. economy with a positive balance of trade, is facing increased competition and the technology base now contains gaps and a budget insufficient to close them. During this same period, investments in France, Italy, Germany, Japan, China, the Soviet Union and other countries have increased greatly. What was once an unquestioned area of U.S. preeminence -- the civil space program -- is now increasingly a field with shared leadership.

TECHNOLOGY READINESS

So many innovative and exciting technological opportunities face us that selecting them becomes a significant challenge in itself. But, the question remains as to

*Associate Administrator
AIAA Fellow

[illegible] will [illegible] rather than requirements, [illegible] that they are [illegible] and will [illegible] and take longer. Worse than not having the technology is to fool ourselves into believing it is ready. If we want to go back to the moon to stay, embark on an expedition to mars, or build the great observatories, we must develop the technology.

The reality of the situation is that the technologies for our future missions are not where they ought to be, nor were they where they should have been to support current programs. Consider the evidence. The Space Shuttle Main Engine development outpaced the technology program for reusable engines that should have supported it....technology to provide options to the solid rocket motor has not been verified....the filament wound solid rocket motor case did not have a basic engineering data base in hand before development was initiated....we find ourselves dependent on RTG's as a power source with no suitable back up....we face an information lag and a data glut (SEASAT operated for only 3 months and it took eight years to analyze the data). In the Space Station era, the data generated will exceed 10 terabits per day, greater than all the written information in the Library of Congress....The SP-100 space nuclear power program had to take a conservative approach to the thermal conversion system because of unacceptable risk with more efficient conversion systems....the technology is not ready for the developement of new launch vehicles....spacecraft payload capacity is unnecessarily limited because we haven't developed the technology to lower the power, propulsion, and structural mass fraction....there are parts of the spectrum we can't exploit for remote sensing because we haven't done the instrument development research....and, human exploration is paced by our limited understanding of the effects of space on the human body and of closed [illegible] support systems. The list goes on and on. [illegible] technology is not leading with solutions, it is chasing problems.

[illegible]

NASA has recently embarked on an effort to define the goals, objectives, and program thrusts to guide the future of the Nation's civil space program. It is NASA's intent that this process produce a blueprint to guide the United States to a position of leadership among the spacefaring nations of Earth. In support of this high priority effort, one of the nine key recommendations of the reorganization plan for NASA is directed at pioneering new technology to ensure that our country has a robust, balanced, and safe space program.

The NCOS recommendations are congruous with two other studies supporting the need for increased research and technology investments in space. A three-year Space Science Board study, "Twenty-year Forecast of Space Science, 1995-2015," provides a logical extension of the Nation's current science program with a balance between astrophysics, planetary exploration, earth science, and life science. The second, a joint DoD NASA study, identifies new space transportation vehicle options required to meet national needs in response to a National Security Study Directive on space transportation. In total, three major studies, the Challenger tradgedy, and foreign competition are transcribed into a bold commitment to the future, a renewed commitment to safety, and the restoring of our technical capability.

A FIRST STEP

To begin the task of strengthening our space technology base, NASA has developed an immediate first step for FY88 to be followed by a second initiative that will enable bolder missions. They are, The Civil Space Technology Initiative (CSTI) and Project Pathfinder.

Transportation

CSTI includes research in technologies for safe and efficient access to space, for earth orbiting operations, and for science supporting technologies (see Fig. 1). In transportation, launch vehicle propulsion will be aimed at reusable, high-performance, LOX hydrogen and LOX hydrocarbon engines for next-generation earth-to-orbit vehicles. A new booster technology program will develop alternatives to the solid rocket motor for future transportation. Technology development will be followed by a series of firing tests to verify performance predictions of pressure-fed liquid rocket propulsion and large-scale hybrid systems. An aerobraking flight experiment will demonstrate how to use the atmosphere to slow an orbital transfer vehicle returning to earth orbit at high velocities. Application of the aerobrake concept to the OTV has the potential for increasing the payload delivered to geosynchronous orbit and return by as much as a factor of two. In the flight experiment, a highly instrumented test article will be deployed from the shuttle and accelerated into the atmosphere and back to low-earth orbit for shuttle recovery.

Operations

Earth-orbiting operations capability will be enhanced through a continued and expanded effort in automation and robotics. The objective of this program is to exploit the potential of artificial intelligence and telerobotics to increase the capability, flexibility, and safety of space- and ground-operations while decreasing associated costs. The NCOS recommended a "special emphasis on intelligent autonomous systems." This CSTI element responds directly to the NCOS recommended acceleration of work to support complex, automated, remote operations. Research in autonomous systems to enable the control of multiple subsystems, including the capability for reasoning and recovery from failures, will be demonstrated with the technology flowing into the Space Station test beds. Launch system software development will also be addressed to begin replacing as much of the routine, labor intensive activities as possible.

A robotics program will demonstrate a multi-arm, highly autonomous capability for remote assembly, repair and servicing of space vehicles. This program is built upon

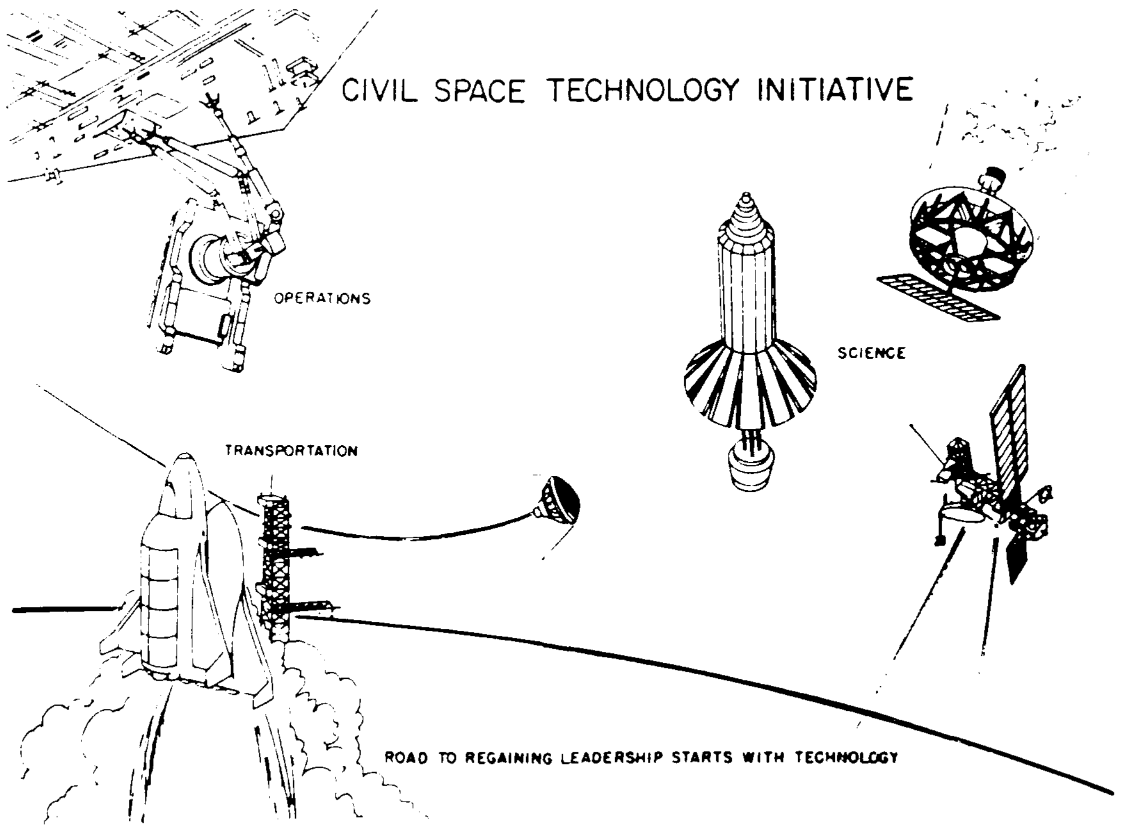

CIVIL SPACE TECHNOLOGY INITIATIVE
OPERATIONS
SCIENCE
TRANSPORTATION
ROAD TO REGAINING LEADERSHIP STARTS WITH TECHNOLOGY

FIGURE 1

research and technology development in five core areas: sensing and perception, control execution, task planning and reasoning, operator interface, and system architecture and integration.

Also in earth-orbiting operations is the requirement for high capacity power systems. Ongoing research will continue on the high power Stirling cycle dynamic thermal-to-electric power conversion system. Performance capabilities are aimed at a five-fold increase in the amount of electric power that can be generated from a reactor of the size being developed in the SP-100 program and are aimed at a doubling of the power-to-weight ratio.

Science

In science supporting technology, emphasis will be on an augmented large flexible structures and control program, and on sensors and data. Future space missions will require very light, very large, flexible structures for large communications satellites, science observing platforms, advanced Space Station configurations, a large deployable reflector, and large baseline interferometers. The structural dynamics and control response of such flexible structures is not well understood. An added problem is the degree of precision attainable for large space-based reflectors especially in the submillimeter range for astronomical observation. To address this problem, lightweight reflector panels made up of composite materials tailored for space durability and surface accuracy are being developed. Active controls research will be included as an approach to maintain the surface precision and precise alignment of seven 1-to-2 meter wide assembled panels.

The program in sensor devices will be aimed at submillimeter technology for high-resolution earth science and astrophysics instruments at wavelengths outside the limits of current observations; solid-state, tunable lasers for lidar instruments to sense atmospheric constituents; and long-life sub-Kelvin detector cooler technology to cool infra-red detector arrays necessary for imaging in a low-thermal background such as that found looking out into deep space.

In the data area, research will be augmented on high rate, high capacity systems that function as smart, on-board processors primarily for imaging information. A combination of high speed, special purpose processors and high capacity data devices is required if we are to discontinue excessive demands on ground communications. CSTI data research also includes verification of a read/write high capacity (a trillion bits) optical disk recorder. This unit would be useful for storing images for overlays and comparisons.

THE SECOND STEP

If CSTI is characterized as a transportation, earth orbit, and science supporting focus, then the second increment, which we call Pathfinder, looks beyond Space Station, beyond LEO, and addresses common technologies that support a range of future missions including a return to the Moon, missions to Mars, and expanded exploration of the outer planets. Pathfinder includes technology thrusts to enable long-term human presence in space, processing of non-terrestrial materials, an autonomous rover, interplanetary transportation, aerobraking and atmospheric capture for planetary return, precision aero-recovery, tether applications, and on-orbit propellant depots for manufacturing and storage of propellants.

A NEW BEGINNING

A new, Agency-wide attitude toward research and technology development is emerging. The past year has been a year of reflecting upon the state of NASA and reconstituting the Shuttle. We should learn from it, be stronger for it, but let it be a turning point for a renewed dedication to the civil space program -- a new beginning. CSTI and Pathfinder are right for NASA and the Nation, and these efforts will challenge and encourage the Agency, industry, and the University community. And, for universities in the space engineering disciplines, an entirely new and important role may be emerging in the scope and manner in which NASA creates university centers for space technology research.

The Nation's technology program fortells what we will be doing twenty years from now. If we follow through on what we have started, the picture of the 21st Century has certainly become brighter.

AIAA-87-1697 Computer Architecture for Future Spacecraft

J. B. Dennis, Massachusetts Institute of Technology, Cambridge, MA

Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program

March 9-11, 1987/Arlington, VA

Computer Architecture for Future Spacecraft

Jack B Dennis

Laboratory for Computer Science
Massachusetts Institute of Technology
545 Technology Square, Cambridge, MA 02139

In the design of computers for spacecraft there is extraordinary emphasis on low weight, small size, and minimum power consumption. These considerations mandate refined packaging and cooling technology, and use of a logic technology that achieves the highest computation rate for given device density and power dissipation—consistent with survivability in space. The goals have been more than sufficiently met, and computers have made possible many impressive achievements in space.

Advances in memory and logic devices have led people to envision that much higher levels of function and intelligence can be built into future spacecraft within the limits on weight, size and power. Authors foresee autonomous vehicles guided by knowledge-based software systems and elaborate vision systems. The motivation is to make the best use of the tiny signal bandwidth available for communication with mother earth.

Nevertheless, serious problems must be addressed if this potential is to yield practical achievements in space, especially where the advanced computing capability is to be applied to mission critical or real time tasks.

Dependability: One problem area is dependability. Significant reliability goals have been met in past space missions. spacecraft have survived deep space missions requiring dependable performance for many years—a truly remarkable feat. This has been accomplished through the use of redundancy, fault detection, and self repair using standby spare components to make space computers tolerant of any single point failure.

The needed dependability has been feasible because software functions are kept simple and critical software components are subjected to unusually thorough checking and verification during development. This thoroughness and the special requirements often imposed on software design by fault tolerance schemes place severe limits on the complexity of software functions that may be undertaken. As tasks are given to the spacecraft computer that require more sophisticated software structures, the difficulty of achieving the needed dependability will increase.

Programmability: The most significant area, however, is programmability. While present architectures for spacecraft computers may be a reasonable choice for the software requirements of present space missions, they are a poor match to those foreseen in the future. Successful application of artificial intelligence methods and the effective development of reusable software parts will require significant architectural innovation if the potential of these techniques is to be realized in the construction of very large software systems for the space environment.

One requirement of the knowledge-based approached to implementing intelligent behavior is a large memory. If the memory of a spacecraft computer is entirely in the form of static semiconductor devices, then size and power limitations will severely limit the level of intelligent behavior possible. The principal technologies available for large spacecraft memories are dynamic semiconductor devices and magnetic bubble memories. High-capacity dynamic memories are very susceptible to soft errors and other forms of failure from radiation exposure, and they have a relatively slow access time. Bubble memories, while resistant to radiation, have such long access times that a hierarchical memory organization will be required.

These facts complicate the design of computers having storage systems to support modular, knowledge-based software [6, 7, 3, 2]. This is because the advanced software desired requires automatic management of the memory system including recognition and recovery of storage occupied by data no longer of relevance. The schemes adopted for fault detection, self repair and software recovery must be consistent with these requirements. Specifically, if the fault tolerance schemes are not transparent to the sensitive application software, then these schemes will impose intolerable restrictions on the software design. I am not personally aware of work that addresses these issues in a sufficiently general way. The achievement of fault tolerance in an advanced programming environment is an unmet challenge to computer system architects.

Dataflow Technology and its Role in Space: Present and future spacecraft computers will continue to have redundant elements to meet dependability requirements. They will also be built of multiple processing elements to obtain the most performance for given weight, size and power. In a dataflow computer each processing element implements a data driven instruction execution mechanism [1].

Weight, Size and Power: Studies of applications show that data flow computers have peak performance capabilities for given weight, size and power well beyond that of other multiprocessor architectures [8, 4]. Moreover these same studies indicate that this peak performance can be closely approached in many important application areas. Thus dataflow computer architecture should be seriously considered for spacecraft applications.

Programmability: The program structures used with dataflow machines are nicely matched to the phrases used to express computations in high level functional programming languages. Hence there is a natural way to split programs into modules that make sense at both the language level and the machine level.

We have recently found that the basic artificial intelligence programming scheme—heuristic search of a tree-structured space of possibilities—can be implemented with extraordinary efficiency on dataflow computers of relatively simple architecture [5]. Thus dataflow computers offer an attractive basis for supporting advanced software for space missions.

Fault Tolerance: Dataflow computers offer good opportunities for applying fault tolerance techniques in a way that is transparent to the application software [10]. For other types of computer systems, this has been achieved rarely and only for restricted classes of high level language programs. The expectation of machines—dataflow computers in particular—that directly implement powerful functional languages makes the vision of space dependable computers performing sophisticated intelligent procedures a realistic vision for the future [9].

References

1. Agerwala, T. and Arvind, Eds. "Special Issue on Data Flow Systems" *Computer 15, 2* (February 1982)

2. Dennis, J. B. On the exchange of information. Proc. of ACM-SIGFIDET Workshop on Data Description and Access. New York, NY, 1970.

3. Dennis, J. B. *Lecture Notes In Computer Science* Volume 30: Modularity. In *Software Engineering: An Advanced Course*, Springer-Verlag, Berlin, Heidelberg, New York, 1975, pp. 128-182.

4. Dennis, J. B. Dataflow computation: a case study. In *Computer Architecture: Concepts and Systems*, V. Milutinovic, Ed., Elsevier, New York, 1987.

5. Dennis, J. B. Dataflow Computation for Artificial Intelligence. In *Supercomputers and AI Machines*, Kai Hwang and Douglas DeGroot, Eds., McGraw-Hill, New York, 1988. To be published.

6. Dennis, J. B. Programming generality, parallelism and computer architecture. In *Information Processing 68*, North-Holland, Amsterdam, 1969, pp. 484-492.

7. Dennis, J. B. On the design and specification of a common base language. Proc. of the Symposium on Computers and Automata, Brooklyn, NY, 1971, pp. 47-74.

8. Dennis, J. B., Gao, G-R., and Todd, K. W. R. "Modeling the weather with a data flow supercomputer". *IEEE Transactions on Computers C-33*, 7 (July 1984), 592-603.

9. Jagannathan, S. Data Backup and Recovery in a Computer Architecture for Functional Programming. MIT/LCS/TR-353, lcs, October, 1985.

10. Leung, C. K.-C. and Dennis, J. B. Design of a fault-tolerant packet communication computer architecture. Tenth Ann. Symp. on Fault Tolerant Computer Systems, IEEE, New York, NY, October, 1980, pp. 328-335.